KINEMATICS, DYNAMICS AND DESIGN OF A SPHERICAL POSITIONING ROBOT FOR SATELLITE TRACKING AND OTHER APPLICATIONS

A THESIS PRESENTED FOR THE DEGREE
OF
DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING
AT THE
UNIVERSITY OF CANTERBURY
by

Timothy Paul Jones

University of Canterbury
1996

A computer-rendered view of the tracking robotic mechanism built at the University of Canterbury.

To Mum and Dad.

Abstract

In order to avoid the keyhole problems associated with the current antenna mounting systems and to meet the requirements of target acquisition and tracking for high gain narrow beamwidth antennas, a novel antenna mounting system has been developed. The antenna is mounted on a parallel robot based on the Phillips-Sherwood mechanism that has either two or three degrees of freedom. Unlike conventional mounts this parallel robotic mounting system is able to point an antenna anywhere in the upper hemisphere without encountering singularities. Three other parallel mechanisms that are derivatives of the Phillips-Sherwood mechanism are analysed, and conclusions are drawn about the feasibility of such mechanisms for beam aiming applications. One mechanism in particular appears to offer many advantages, and a robot based on this mechanism may in fact prove to be superior to the first robotic aiming mechanism developed at the University of Canterbury.

Publications

Dunlop G. R., Lintott A. B. and Jones T. P., "Linear and Spherical Robots with Three DOF", *Proc. First Australian Workshop on the Theory of Machines & Mechanisms*, Melbourne University Press, ISBN 0-7325-0594-1, pp. 175-201, (1992).

Dunlop G. R., Lintott A. B. and Jones T. P., "Three DOF parallel robots for linear and spherical movements", *ISRAM'94, Intelligent Automation and Soft Computing*, Eds. Jamshidi M, and Nguyen CC, TSI Press, ISBN 0-962-7451-4-6, Vol 1, pp. 655-660, (1994).

Dunlop G. R. and Jones T. P., "Three Axis Parallel Link Robot for Key Hole Free Antenna Tracking", Proc. IPENZ Conf., Nelson, Feb. 18-21, Vol. 1, pp. 56-59, (1994).

Dunlop G. R. and Jones T. P., "Gravity Counter Balancing of a Parallel Robot for Antenna Aiming", *ISRAM'96, Sixth International Symposium on Robotics and Manufacturing*, Montpellier, France, May 27-30, (1996).

The following paper has been accepted for publication and is in press.

Dunlop G. R. and Jones T. P., "Position Analysis of a 3 DOF Parallel Manipulator", *Mechanisms and Machine Theory.*

Acknowledgments

I am indebted to my supervisor, Dr Reg Dunlop, for his guidance and encouragement throughout the duration of this project. I also wish to thank Dr Chris Damaren for his assistance with the later part of this work, particularly the dynamics. I am extremely grateful to Andrew Lintott with whom I have worked closely for the best part of five years, and also to Professor Harry McCallion for his contributions to a multitude of problems that have cropped up during the course of this work.

I would like to thank the workshop staff, particularly Mr Scott Amies who spent many frustrating hours building the robot. I am appreciative of the contribution of my fellow postgraduate students, for their ideas and interesting discussions on many topics. The assistance of the academic staff and technicians, Mr Andrew Cree in particular, is acknowledged.

I wish to acknowledge the private communications of Professor Kenneth Hunt and Dr Jack Phillips who have contributed their own thoughts and recommendations on the project.

Finally, I would like to thank Melissa for her encouragement and patience in standing by a not-so-financial partner.

Contents

List of Figures

List of Symbols and Abbreviations

Symbol	Description
A_e	Effective antenna aperture
Az	Azimuth angle
B_n	Points used to describe the position of the spherical joints
$\boldsymbol{B}_n$	Position of the points describing the spherical joints in the principal frame $\mathcal{F}_0$
${}^n\boldsymbol{B}_n$	Position of the points describing the spherical joints in the local frames $\mathcal{F}_n$
${}^n\boldsymbol{c}_n$	First moment of mass of body B_n about O_n
${}^m\mathbf{C}_n$	Rotation matrix relating the components of a vector expressed in frame $\mathcal{F}_n$ to those of the same vector expressed in frame $\mathcal{F}_m$
${}^m\dot{\mathbf{C}}_n$	Time derivative of ${}^m\mathbf{C}_n$
dof	Degree-of-freedom
d_i	Inner diameter of thick walled tube
d_o	Outer diameter of thick walled tube
El	Altitude or elevation angle
E	Young's modulus of elasticity
$\sum f$	Sum of the freedoms
$\mathcal{F}_n$	Frame n
$\boldsymbol{F}_p$	External force acting through the point O_p
$\underline{\boldsymbol{F}}_{n+1,n}$	Force on body B_n due to B_{n+1} at O_{n+1}
${}^{n+1}\boldsymbol{F}_{n+1,n}$	Components of $\underline{\boldsymbol{F}}_{n+1,n}$ expressed in frame $\mathcal{F}_{n+1}$
$\underline{\boldsymbol{F}}_{n\hat{+}1,n}$	Force on body B_n due to B_{n+1} at $O_{n\hat{+}1}$
$\underline{\boldsymbol{F}}_{n-1,n}$	Force on body B_n due to B_{n-1} at O_n
${}^n\boldsymbol{F}_{n-1,n}$	Components of $\underline{\boldsymbol{F}}_{n-1,n}$ expressed in frame $\mathcal{F}_n$
$\underline{\boldsymbol{F}}_{n,T}$	Total force on B_n at O_n
${}^n\boldsymbol{F}_{n,T}$	Components of $\underline{\boldsymbol{F}}_{n,T}$ expressed in frame $\mathcal{F}_n$
$\underline{\boldsymbol{F}}_{n,ext}$	External force on B_n at O_n
${}^n\boldsymbol{F}_{n,ext}$	Components of $\underline{\boldsymbol{F}}_{n,ext}$ expressed in frame $\mathcal{F}_n$
$\boldsymbol{F}_z$	Closed loop actuator (control) forces expressed in z space
$\mathcal{J}_{n-1,n}$	Generalised force on body B_n due to B_{n-1} at O_n
$\mathcal{J}_{Int}$	Assembled system of generalised interbody forces $\mathcal{J}_{n-1,n}$

${}^{n}\mathcal{J}_{C}$	Control forces (forces due to actuators)
$\mathcal{J}_{C}$	Assembled system of control forces ${}^{n}\mathcal{J}_{C}$
${}^{n}\mathcal{J}_{n,con}$	Constraint forces (forces acting on body n due to reactions with body n-1).
$\mathcal{J}_{Con}$	Assembled system of constraint forces ${}^{n}\mathcal{J}_{n,con}$
$\mathcal{J}_{n,T}$	Generalised total force on B_n at O_n
$\mathcal{J}_{T}$	Assembled system of generalised total body forces $\mathcal{J}_{n,T}$
$\mathcal{J}_{n,ext}$	Generalised external force on B_n at O_n
$\mathcal{J}_{Ext}$	Assembled system of generalised external forces $\mathcal{J}_{n,ext}$
$\mathcal{J}_{E}$	Assembled system of generalised external forces premultiplied by $\mathbf{P}^{T}\mathcal{G}^{T}\mathcal{T}^{T}$ i.e. $\mathcal{J}_{E} = \mathbf{P}^{T}\mathcal{G}^{T}\mathcal{T}^{T}\mathcal{J}_{Ext}$
$\mathcal{J}_{Non}$	Column of generalised forces associated with the nonlinear kinematic acceleration term
$\mathcal{J}_{Non,\beta}$	Assembled system of generalised forces associated with the non linear kinematic acceleration term forces premultiplied by $\mathbf{P}^{T}\mathcal{G}^{T}\mathcal{T}^{T}$ i.e. $\mathcal{J}_{Non,\beta} = \mathbf{P}^{T}\mathcal{G}^{T}\mathcal{T}^{T}\mathcal{J}_{non}$
$\underline{G}_{n+1,n}$	Torque about O_{n+1} on B_n due to B_{n+1}
$\underline{G}_{n-1,n}$	Torque about O_n on B_n due to B_{n-1}
$\underline{\boldsymbol{G}}_{n-1,\hat{n}}$	Torque about $O_{\hat{n}}$ on B_n due to B_{n-1}
${}^{n}\boldsymbol{G}_{n-1,\hat{n}}$	Components of $\underline{\boldsymbol{G}}_{n-1,\hat{n}}$ expressed in frame $\mathcal{F}_n$
$\underline{G}_{n\hat{+}1,n}$	Torque about $O_{n\hat{+}1}$ on B_n due to B_{n+1}
${}^{n+1}\boldsymbol{G}_{n\hat{+}1,n}$	Components of $\underline{G}_{n\hat{+}1,n}$ expressed in frame $\mathcal{F}_{n+1}$
$\underline{\boldsymbol{G}}_{n,T}$	Total torque about O_n acting on B_n
${}^{n}\boldsymbol{G}_{n,T}$	Components of $\underline{\boldsymbol{G}}_{n,T}$ expressed in frame $\mathcal{F}_n$
$\underline{\boldsymbol{G}}_{n,ext}$	External torque about O_n acting on B_n
${}^{n}\boldsymbol{G}_{n,ext}$	Components of $\underline{\boldsymbol{G}}_{n,ext}$ expressed in frame $\mathcal{F}_n$
${}^{n}\mathcal{G}_{n,n}$	Transformation from the relative generalised velocity observed in the rotating frame $\mathcal{F}_n$ to the relative generalised velocity observed in $\mathcal{F}_1$ expressed in $\mathcal{F}_n$
${}^{n}\dot{\mathcal{G}}_{n,n}$	Time derivative of ${}^{n}\mathcal{G}_{n,n}$
$\mathcal{G}$	Assembled system of transformation matrices ${}^{n}\mathcal{G}_{n,n}$
$\mathcal{G}_{mc}$	Global (assembled system of) transformation matrices associated with the mechanism only
g	Number of joints
$\mathbf{g}$	Acceleration due to gravity (+ve down)
G	Antenna gain
IFX	Step motor controller made by the Parker Hannifin Corporation
I	Area moment of inertia
${}^{n}\boldsymbol{J}_{n}$	Second moment of mass (moment of inertia) of body B_n about O_n

l_n	Length of the arm n
m_n	Mass of body B_n
$\boldsymbol{M}$	The position of the point midway between O_b and O_p.
${}^{n}\boldsymbol{M}_{n}$	Mass matrix for body B_n expressed in frame $\mathcal{F}_n$
$\mathbf{M}$	Assembled system of mass matrices ${}^{n}\boldsymbol{M}_{n}$
$\mathbf{M}_{\beta\beta}$	Assembled mass matrix $\mathbf{M}$ premultiplied by $\mathbf{P}^T\mathcal{G}^T\mathcal{T}^T$ and post-multiplied by $\mathcal{T}\mathcal{G}\mathbf{P}$ i.e. $\mathbf{M}_{\beta\beta} = \mathbf{P}^T\mathcal{G}^T\mathcal{T}^T\mathbf{M}\mathcal{T}\mathcal{G}\mathbf{P}$
$\boldsymbol{M}_P$	External moment acting about the point O_p
${}^{m}\boldsymbol{M}_{P_n}$	Moment about the point P_n expressed in the $\mathcal{F}_m$ frame
$\boldsymbol{N}_n$	Spherical joint force reaction force
n	Number of links
O_b	Point used to describe the centroid of the base
$\boldsymbol{O}_b$	Position of the point used to describe the centroid of the base in the principal frame $\mathcal{F}_0$
O_p	Point used to describe the centroid of the platform
$\boldsymbol{O}_p$	Position of the point used to describe the centroid of the platform in the principal frame $\mathcal{F}_0$
O_I	Origin of inertial frame $\mathcal{F}_I$
O_n	Location of attachment point on body B_n for joint connecting body B_n to body B_{n-1}
O_{n+1}	Location of attachment point on body B_n for joint connecting body B_n to body B_{n+1}
$\underline{\boldsymbol{p}}_{n,n+1}$	Position vector from O_n to O_{n+1}
${}^{n}\boldsymbol{p}_{n,n+1}$	Components of $\underline{\boldsymbol{p}}_{n,n+1}$ expressed in frame $\mathcal{F}_n$
$\dot{\underline{\boldsymbol{p}}}_{n,n+1}$	Time derivative of $\underline{\boldsymbol{p}}_{n,n+1}$ w.r.t. the inertial frame $\mathcal{F}_I$
$\overset{o}{\underline{\boldsymbol{p}}}_{n,n+1}$	Time derivative of $\underline{\boldsymbol{p}}_{n,n+1}$ w.r.t. the frame $\mathcal{F}_n$
$\underline{\boldsymbol{p}_{c_n}}$	Position of the centre of mass of B_n relative to O_n
${}^{n}\boldsymbol{p}_{c_n}$	Components of $\underline{\boldsymbol{p}_{c_n}}$ expressed in frame $\mathcal{F}_n$
P_n	Points used to describe the revolute joints
$\boldsymbol{P}$	The position of the tracked object
$\mathbf{P}_n$	Projection matrix for the nth joint
$\mathbf{P}$	Assembled system of projection matrices $\mathbf{P}_n$
$\mathbf{P}_{mc}$	Global (assembled system of) projection matrices associated with the mechanism only
$\mathbf{Q}_n$	Projection matrix mapping the constraint forces ${}^{n}\mathcal{T}_{n,con}$ onto specified axes
$\mathbf{Q}$	Assembled system of projection matrices $\mathbf{Q}_n$
R	Radius of the antenna
${}^{n+1}\mathbf{R}_n$	Displacement equivalence matrix relating frame $\mathcal{F}_n$ to frame $\mathcal{F}_{n+1}$

${}^{n+1}\dot{\mathbf{R}}_n$	Time derivative of ${}^{n+1}\mathbf{R}_n$
$\mathbf{R}_{mc}$	Projection matrix used to project the independent rates into the global (assembled system of) rates of the mechanism only
$\mathbf{R}$	Projection matrix used to project the independent rates into the global (assembled system of) rates pertaining to the ship, stabilised platform and mechanism
$\dot{s}_{\hat{A}A}$	Array of rates describing the surge, sway and heave of the ship
$\underline{s}_{n\hat{+}1,n+1}$	Relative displacement of O_{n+1} on body B_{n+1} w.r.t. $O_{n\hat{+}1}$ on body B_n
${}^{n+1}s_{n\hat{+}1,n+1}$	Components of ${}^{n+1}s_{n\hat{+}1,n+1}$ expressed in frame $\mathcal{F}_{n+1}$
$\dot{\underline{s}}_{n\hat{+}1,n+1}$	Time derivative of $\underline{s}_{n\hat{+}1,n+1}$ with respect to frame $\mathcal{F}_{n+1}$
${}^{m}\mathbf{S}_n$	Euler rate projection matrix
${}^{m}\mathbf{T}_n$	Homogeneous transformation matrix, transforming from a point expressed in the $\mathcal{F}_n$ frame to the $\mathcal{F}_m$ frame.
${}^{m}\mathcal{J}_n$	Interbody transformation from B_n to B_m
${}^{m}\dot{\mathcal{J}}_n$	Time derivative of ${}^{m}\mathcal{J}_n$
$\mathcal{J}$	Assembled system of interbody transformation matrices ${}^{m}\mathcal{J}_n$
$\mathcal{J}_{mc}$	Global (assembled system of) interbody transformation matrices associated with the mechanism only
VGT	Variable geometry truss
${}^{n+1}\mathbf{V}^{\times}_{n+1,n}$	Velocity matrix obtained from the derivative of displacement equivalence matrix ${}^{n+1}\mathbf{R}_n$, expressed in frame $\mathcal{F}_{n+1}$
$\underline{\mathbf{v}}_n$	Absolute translational velocity of O_n on body B_n
${}^{n}\mathbf{v}_n$	Components of $\underline{\mathbf{v}}_n$ expressed in frame $\mathcal{F}_n$
$\underline{\mathbf{v}}_{n+1,n\hat{+}1}$	Relative velocity of O_{n+1} on body B_{n+1} w.r.t. $O_{n\hat{+}1}$ on body B_n as observed in frame $\mathcal{F}_I$ i.e. $\underline{\mathbf{v}}_{n+1,n\hat{+}1} = \dot{\underline{s}}_{n\hat{+}1,n+1}$
${}^{n+1}\mathbf{v}_{n+1,n\hat{+}1}$	Components of $\underline{\mathbf{v}}_{n+1,n\hat{+}1}$ expressed in frame $\mathcal{F}_{n+1}$
${}^{n+1}\dot{v}_{n+1,n\hat{+}1}$	Time derivative of ${}^{n+1}\mathbf{v}_{n+1,n\hat{+}1}$ w.r.t. the inertial frame $\mathcal{F}_I$
$\underline{\mathbf{v}}_{On\hat{+}1}$	Absolute velocity of $O_{n\hat{+}1}$ on body B_n
${}^{n}v_n$	Generalised velocity of B_n w.r.t. $\mathcal{F}_I$ expressed in frame $\mathcal{F}_n$
v	Assembled system of generalised velocities ${}^{n}v_n$
${}^{n}v_{n,\hat{n}}$	Generalised relative velocity of O_{n+1} on body B_{n+1} w.r.t. $O_{n\hat{+}1}$ on body B_n as observed in frame $\mathcal{F}_I$
$\hat{v}$	Assembled system of relative velocities ${}^{n}v_{n,\hat{n}}$
${}^{n}v_{m,C}$	Generalised velocity of body m w.r.t. $\mathcal{F}_C$ expressed in frame $\mathcal{F}_n$
v_{mc}	Global (assembled system of) generalised velocities associated with the mechanism only w.r.t. the moving ship's frame $\mathcal{F}_C$
$\underline{\omega}_n$	Absolute angular velocity of frame $\mathcal{F}_n$
${}^{n}\omega_n$	Components of $\underline{\omega}_n$ expressed in frame $\mathcal{F}_n$

$\underline{\omega}_{n+1,n}$ Angular velocity of frame $\mathcal{F}_{n+1}$ relative to frame $\mathcal{F}_n$

${}^{n+1}\omega_{n+1,n}$ Components of $\underline{\omega}_{n+1,n}$ expressed in frame $\mathcal{F}_{n+1}$

$\underline{\omega}_{n+1,n\hat{+}1}$ Relative angular velocity of frame $\mathcal{F}_{n+1}$ w.r.t. frame $\mathcal{F}_{n\hat{+}1}$

${}^{n+1}\omega_{n+1,n\hat{+}1}$ Components of $\underline{\omega}_{n+1,n\hat{+}1}$ expressed in frame $\mathcal{F}_{n+1}$

${}^{n+1}\mathbf{w}_{n+1,n\hat{+}1}$ Relative velocity of O_{n+1} on body B_{n+1} w.r.t. $O_{n\hat{+}1}$ on body B_n as observed in frame $\mathcal{F}_{n+1}$ expressed in frame $\mathcal{F}_{n+1}$

$${}^{n+1}\mathbf{w}_{n+1,n\hat{+}1} = {}^{n+1}\dot{\mathbf{s}}_{n\hat{+}1,n+1} \equiv \mathcal{F}_{n+1}\dot{\underline{s}}_{n\hat{+}1,n+1}.$$

w.r.t. With respect to

w Deflection

$\mathbf{X}_{mc}$ Matrix enforcing velocity constraints of the mechanism only

$\mathbf{X}$ Matrix enforcing velocity constraints pertaining to the ship, stabilised platform and mechanism

$\mathbf{Y}_{mc}$ Matrix enforcing the rate constraints of the mechanism only

$\mathbf{Y}$ Matrix enforcing the rate constraints pertaining to the ship, stabilised platform and mechanism

y Distance from N.A to outer fibre

$\dot{\boldsymbol{z}}$ Array of independent rates

$\ddot{\boldsymbol{z}}$ Array of independent time derivatives of rates

β_n Arm angles

$\dot{\beta}_n$ Rate of the joint n

$\dot{\boldsymbol{\beta}}$ Assembled system of rates $\dot{\beta}_n$

$\dot{\boldsymbol{\beta}}_{mc}$ Global (assembled system of) rates associated with the mechanism only

$\ddot{\beta}_n$ Time derivative of the rate of the joint n

$\ddot{\boldsymbol{\beta}}$ Assembled system of rate time derivatives $\ddot{\beta}_n$

$\ddot{\boldsymbol{\beta}}_{mc}$ Global (assembled system of) rates time derivatives associated with the mechanism only

$\boldsymbol{\vartheta}$ Array of Euler angles describing the roll, pitch and yaw of the stabilised platform

θ_{bw} Antenna beamwidth

θ_2 Euler angle describing the orientation of the platform relative to the frame $\mathcal{F}_0$, note $\theta_2 = \varphi_2$

θ_{3a} Euler angle describing the orientation of the platform relative to the frame $\mathcal{F}_0$, note $\theta_{3a} = \varphi_{3a}$

θ_{3b} Euler angle describing the orientation of the platform relative to the frame $\mathcal{F}_0$

φ_2 Euler angle describing the aiming coordinates of the tracked object relative to the ship's frame $\mathcal{F}_C$

φ_{3a} Euler angle describing the aiming coordinates of the tracked object relative to the ship's frame $\mathcal{F}_C$

φ_{3b}	Euler angle describing the aiming coordinates of the tracked object relative to the ship's frame $\mathcal{F}_C$
λ	Wave length of the received radio wave
$\underline{\lambda}$	Constraint forces produced by virtual cut in the parallel mechanism
$\underline{\rho}_{n,n+1}$	Position vector from O_n to O_{n+1}
${}^{n}\rho_{n,n+1}$	Components of $\underline{\rho}_{n,n+1}$ expressed in frame $\mathcal{F}_n$
$\dot{\underline{\rho}}_{n,n+1}$	Time derivative of $\underline{\rho}_{n,n+1}$ w.r.t. the inertial frame $\mathcal{F}_I$
$\overset{o}{\underline{\rho}}_{n,n+1}$	Time derivative of $\underline{\rho}_{n,n+1}$ w.r.t. the frame $\mathcal{F}_n$
${}^{n}\dot{\rho}_{n,n+1}$	Components of $\overset{o}{\underline{\rho}}_{n,n+1}$ expressed in frame $\mathcal{F}_n$
σ_b	Stress due to bending
ϕ_2	Euler angle describing the aiming coordinates of the tracked object relative to the inertial frame $\mathcal{F}_I$
ϕ_{3a}	Euler angle describing the aiming coordinates of the tracked object relative to the inertial frame $\mathcal{F}_I$
ϕ_{3b}	Euler angle describing the aiming coordinates of the tracked object relative to the inertial frame $\mathcal{F}_I$
ψ	Array of Euler angles describing the roll, pitch and yaw of the ship
${}^{n+1}\Omega^{\times}_{n+1,n}$	Angular velocity matrix obtained from the differential equation for the rotation matrix ${}^{n+1}\mathbf{C}_n$, expressed in $\mathcal{F}_{n+1}$
$\mathbf{1}$	Identity matrix

Chapter 1

Introduction to satellite-tracking mounting systems

1.1 Introduction

Satellite-tracking systems are employed to track weather and earth resources satellites, space shuttles and unmanned space probes. An earth station that transmits or receives signals from a satellite is an important link in the communications system. The rate at which data can be reliably received is determined by the signal to noise ratio (S/N) of the antenna dish and receiver. The signal power at the receiver can be increased in two ways. First, the transmitted power from the satellite can be increased as is done for direct broadcasting satellites, and second, the amount of received signal energy can be increased by collecting it over a larger area. Large power sources which operate for many years are expensive when launched into orbit. Hence limited power is used for most earth-monitoring satellites.

The second option of increasing the received energy by collecting it over a larger area is described in Nathamson [1969] by the antenna gain equation

$$G = \frac{4\pi A_e}{\lambda^2} \tag{1.1}$$

where G is the antenna gain, A_e is the effective antenna aperture which is the actual area πR^2 times the antenna energy collecting efficiency, and λ is the wave length of the received radio wave. The antenna beamwidth θ_{bw} is given by

$$\theta_{bw} = \frac{\lambda^2}{A_e}. \tag{1.2}$$

Combining equations (1.1) and (1.2) we get

$$G = \frac{4\pi}{\theta_{bw}}. \tag{1.3}$$

Thus the requirement to increase S/N by increasing A_e gives an increase in the antenna gain and a corresponding decrease in the antenna beamwidth. The decreased antenna beam width is the cause of the keyhole problem. With a large beamwidth the limitations of conventional antenna mounts do not affect reception. With a narrow beamwidth the antenna is larger, hence the supporting mount structure is much more massive and so the mount limitations affect antenna operation over some regions of the visible hemisphere.

In the following sections, the current alt-azimuth and X-Y antenna mounting systems are described, together with their corresponding keyholes. Discussion is extended to astronomical mounts, and to multi axis mounts used in marine environments. A six degree of freedom parallel mechanism is briefly described which is capable of solving the keyhole problem.

Finally, a two dof and three dof parallel mechanism are introduced; either of these is able to circumvent the keyhole problem. It is these two mechanisms and their derivatives which are the research topics of this thesis.

1.2 Conventional earth bound mounting systems

The track followed by an orbiting satellite is approximately elliptical in shape. Communications satellites in a geostationary orbit above the equator rotate at the same rate as the earth so that they appear almost stationary when viewed from the earth's surface. Small perturbations occur due to the non homogeneous nature of the earth as well as the gravitational attraction of bodies other than earth. The azimuth (Az) and altitude (Alt) or elevation (El) angles of the satellite viewed from the earth's surface (c.f. figure 1.1) are also known as the look angles of the satellite. Azimuth is measured eastward from geographic north to the projection of the satellite path on to the horizontal plane at the earth station. Elevation is the angle measured above the horizontal plane to the look path. The antenna mounting is used to aim the antenna along the look angles so that the satellite is within the beam of the antenna. Ideally, satellite tracking requires a maximum of two degrees of freedom. There are three standard two dof mounting methods for achieving this. These are the Alt-Az, the X-Y, and the astronomical equatorial mounting systems.

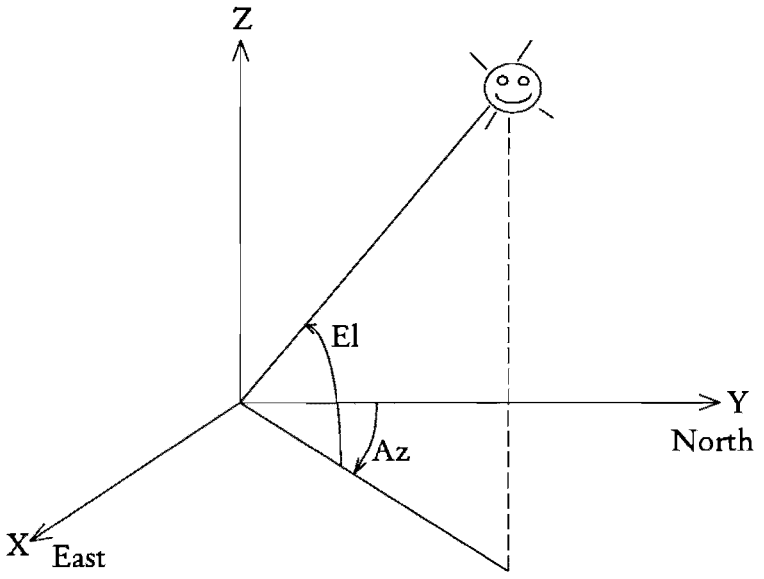


Figure 1.1 Look angles.

1.2.1 Alt-azimuth mountings and the keyhole

The standard alt-azimuth mount consists of a horizontal axis revolute joint which is attached to the rear of the antenna as shown in figure 1.2. The look angles are set by rotating the vertical joint through the azimuth angle from the North, and then rotating the horizontal joint through the elevation angle from the horizon. If a moving satellite is tracked through the zenith or very close to it, then when the elevation reaches almost 90° the azimuth has to rotate through 180°. Thus, the system has a singularity about the zenith. During the time taken for the 180° azimuth rotation, the station can lose contact with the satellite. This is known as the keyhole problem. It occurs when the satellite track is through a region (the keyhole) around the zenith which requires a large change in one of the positioning coordinates for a small change in the satellite position. Dynamic positioning errors allow the satellite to move out of the antenna beam and contact is broken.

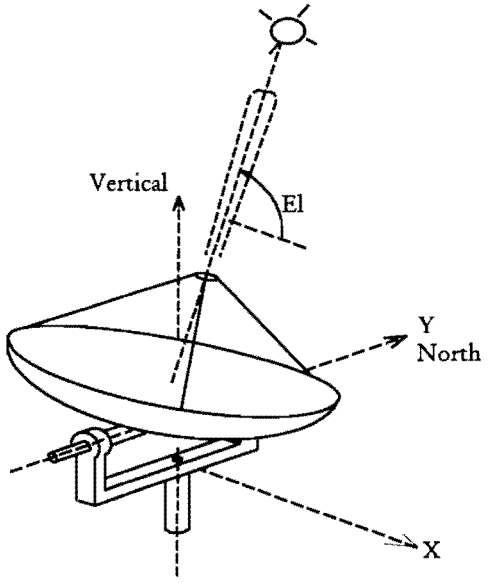


Figure 1.2 Alt-Azimuth mount.

The problem becomes particularly severe when the tracking system is mounted on a ship. The rolling and pitching action of a ship causes the singularity of the alt-azimuth mount to trace out an elliptical shaped conical region. Communication within this region will be unreliable due to the effective increase in size of the keyhole. Similar problems can also arise with wind loading on large antennas.

1.2.2 X-Y mountings and associated keyholes

In order to overcome the effect of having a keyhole about the vertical axis, an X-Y mounting system can be used (c.f. figure 1.3).

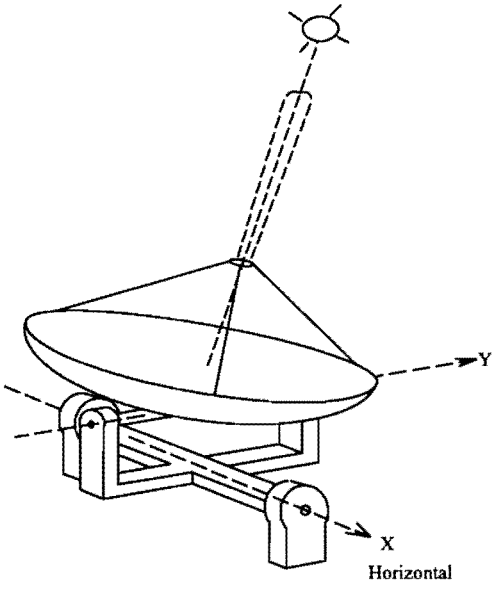


Figure 1.3 X-Y mount.

The X-Y mount consists of two orthogonal axes: a horizontal axis revolute joint carries a second perpendicular axis revolute joint which is attached to the rear of the antenna dish. Each look angle affects both control angles in the X-Y mount; thus the axes are not decoupled and control is more complex. The advantage of the X-Y mount is that it does not have a keyhole about the vertical, but it does have two other keyholes, one at each end of the horizontal axis.

If the horizontal x axis is cantilevered from one end then the mount is the same as the alt-azimuth mount laid on its side with the vertical axis moved to the x axis. The keyholes remain in the same position because of dynamic limitations.

When receiving signals from deep space probes, there is only one chance of capturing the data sent by the space craft and a loss of contact is not acceptable. In the case of the NASA deep space exploration antennas, two X-Y mounted antennas (10m and 30m) are mounted at each of the three global receiving sites. At each site, the horizontal axes of the two antennas are orthogonal. Thus the keyholes of each antenna are covered by the other antenna. This is an effective but expensive solution to the keyhole problem.

1.2.3 Astronomical mountings and associated keyholes

The astronomical mount is similar to the alt-azimuth mount, but instead of having the first axis vertical, it is tilted so that the axis is parallel to the rotation axis of the earth. This particular astronomical arrangement uses the equatorial coordinates of right ascension and declination with the right ascension axis moving one revolution each 24 hours so that the telescope or antenna remains locked on a star. The keyholes are then along the North-South axis. This type of mounting system was used for radio telescopes and for the original satellite-tracking antennas at the NASA Goldstone site. It was also used for the MARS1 installation which is now a US national monument.

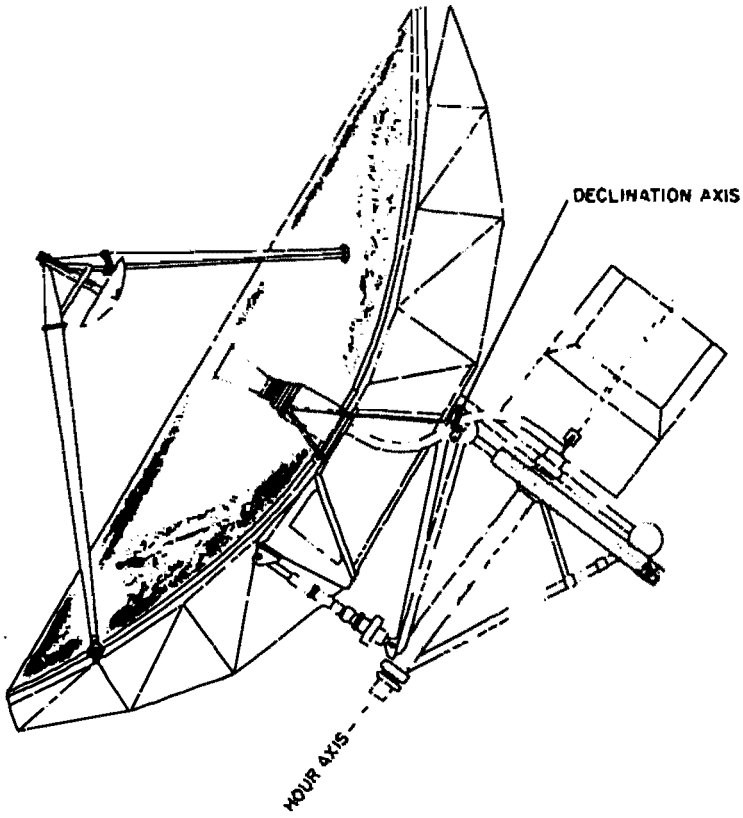


Figure 1.4 Astronomical mount.

1.3 Trajectory optimisation to minimise pointing error

Crawford and Brush [1995] anticipate the trajectory close to the zenith in order to minimise the pointing error. Using the known characteristics of the servo-mechanism, a modification to the antenna trajectory can be implemented so as to minimise the impact of the servo system torque

limits on the pointing accuracy. The two basic approaches to minimising the pointing error are briefly described below.

1) A trajectory is chosen so that the peak azimuth angular error is kept to less than the antenna beam width while the azimuth servo races (leads) the satellite pointing demand. When the satellite tracking mechanism is attempting to lead the satellite trajectory, the high velocity and acceleration demand can be anticipated since the path of the satellite is known in advance. Once the point of closest approach to the zenith has been passed, the antenna follows the satellite as the azimuth demand decreases. This approach enables the large azimuth rotation to occur over a longer period of time (double), thus decreasing the speed and acceleration demand on the azimuth servo.

2) The second approach is to choose a trajectory that will take the antenna 'over the top' (past 90° elevation) with minimum pointing error. Provided the satellite passes within a beam width of the zenith, large rapid azimuth rotations can be avoided.

These solutions are only possible if there is prior knowledge of the satellite trajectory and if the pointing error is less than the beam width. It is applicable to ground based stations employing small tracking antennas (i.e. the beam width is not too small and the gain is low). These solutions have been successfully simulated by Crawford and Brush who were able to keep the pointing error of the tracking system well within the antenna beam width for a NOAA 11 weather satellite.

1.4 Multi-axis mountings for marine applications and keyholes

To overcome the limitations of a two axis mount, extra axes are introduced in the multi-axis antenna systems. The operation of these multi-axis mounting systems is discussed below. The introduction of a third or fourth degree of freedom to overcome the limitations of a two axis mount in the vicinity of a keyhole is quite common in marine communication mounting systems [c.f. Johnson 1978], and in earth resource and weather satellite tracking systems. However, the increased mechanical complexity is costly and the axes must be computer controlled so that the mechanical singularities which cause the keyhole can be avoided. The tracking system key holes are avoided by taking advantage of the extra degrees of freedom to steer the mounting mechanism away from these singularities. Also, the connecting revolute joints must be very stiff so as to avoid the cumulative errors associated with serial mechanisms.

Three axis stabilisation: Cross-elevation over elevation over azimuth

This arrangement is shown in figure 1.5. Two gyros are used to stabilise the two x-y axes of the antenna at all attitudes. When the satellite is near the zenith the cross-elevation axis takes out roll without requiring rapid azimuth axis rotation. When the satellite is near the horizon the cross-elevation axis is parallel with the azimuth axis and takes out short term compass errors and the geometrical error of the azimuth axis. Typical pointing errors when applied with active stabilisation are less than ±3° under typical maritime conditions [c.f. CCIR report, 1978].

Four axis stabilisation

The four axis arrangement shown in figure 6 enables stabilisation and pointing problems to be decoupled.

Active stabilisation

This method of stabilisation uses a reference stabilisation unit consisting of gyroscopes and level sensors to sense the ship's motion. The reference unit generates signals which drive the

power servos to control each mounting axis. Pointing errors are about ±0.5° in a typical marine environment.

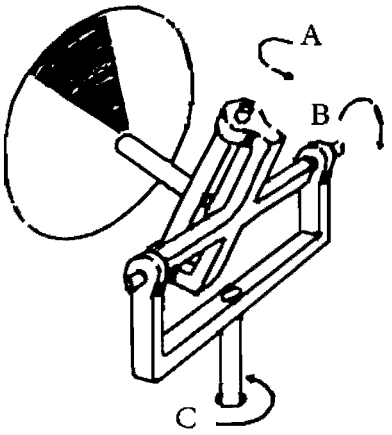


Figure 1.5 Three axis stabilisation. Cross-elevation over elevation over azimuth.

Passive stabilisation with compound pendulums

In this form of stabilisation, the inertia of a compound pendulum is used to stabilise a platform on which the antenna is mounted. The period of the pendulum is much higher than the roll period of the ship. This method is discussed in Kirby [1973] and is simple and cheap but errors can be as high as ±6° which is unacceptable for a high gain antenna.

Passive stabilisation with flywheels

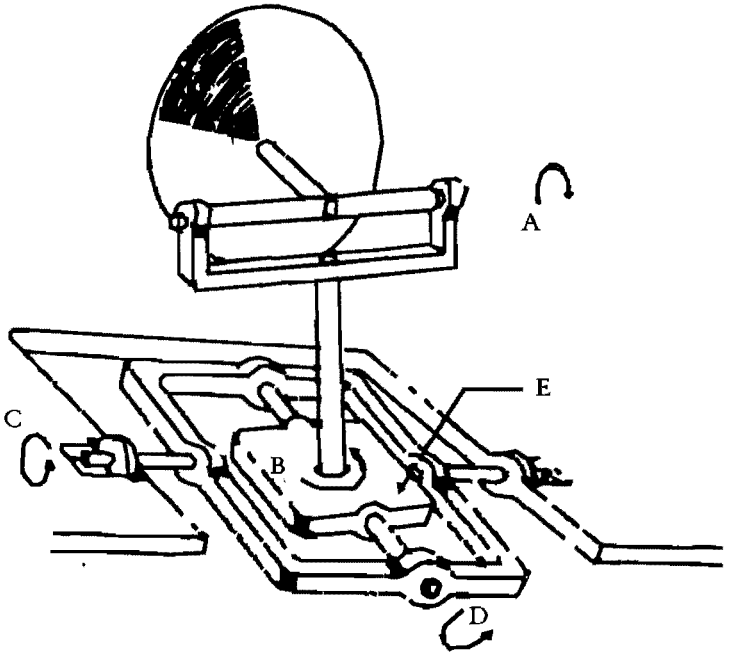


Figure 1.6 Four axis stabilisation. Elevation over azimuth on a two axis gimbal.

This method shown in figure 1.6 uses two contra rotating flywheels to provide an opposing torque if the antenna mount moves. Compared with compound pendulums a greater moment of inertia can be obtained for a given increase in mass. Therefore, bearing friction has less effect and the centre of gravity can be closer to the axis. The disadvantage of this system is that the mounts employing stabilisation tend to be larger and heavier than those used in active stabilisation methods. Accuracies are similar to those that can be achieved with active stabilisation.

1.5 Alternative mounting systems

The problems associated with conventional mounts sparked a search for an alternative mechanism that has two dof and has no singularities in the upper hemisphere. These alternative mounts are based on parallel mechanisms that are able to overcome the keyhole problem.

1.5.1 Six axis parallel mechanism

The use of a parallel robot mechanism to aim an antenna was first suggested by Fichter and McDowel [1980] but not as a method for providing full hemispherical coverage without keyhole problems. The use of this mechanism to overcome the keyhole problem associated with earth resource data receiving stations was proposed by Ellis [1987] and extended to the more severe keyhole problems associated with high bandwidth marine satellite communication by Dunlop & Afzulpurkar [1988].

Figure 1.7 Stewart Platform based antenna mount.

Dunlop & Afzulpurkar used a six degree of freedom mechanism (c.f. figure 1.7). This type of mounting is named after Stewart [1965] and is commonly used as a flight simulator mounting system. The mechanism uses parallel linkages and is able to achieve the required hemispherical coverage for a general purpose antenna mounting system. The six axis positioning system contains eight singularities in the upper hemisphere. Satellite tracking needs only two degrees of freedom so the remaining four degrees of freedom available in the six link parallel mechanism are used to avoid these singularities and to maximise stiffness with respect to the look angles.

This six axis mechanism was shown by Dunlop & Afzulpurkar to be fully capable of tracking an orbiting satellite [cf. Afzulpurkar 1990]. The entire antenna mounting system could be produced at a cost less than a conventional three or four axis tracking system [c.f. Dunlop 1989] such as those discussed by Miya [1981], and in the CCIR report [1978]. If there are any limitations to this system, it is because it is a full six degree-of-freedom system and only two degrees of freedom are required for satellite tracking. Therefore four degrees of freedom serve only to avoid the singularities associated with this type of linkage.

1.5.2 Three axis parallel mechanism

A three degree-of-freedom robot based on a linkage described by Hunt [1973,1978] can be used as an antenna driving mechanism (c.f. figure 1.8).

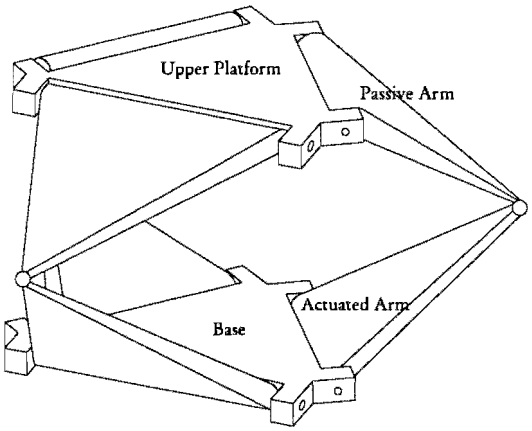


Figure 1.8 Three axis parallel link mechanism.

In a private communication, Hunt attributes the original mechanism to a 1968 development by Phillips and Sherwood. Phillips [1984] published pictures of the three dof mechanism for use as a constant velocity joint. Although the mechanism has been patented by Lambert [1987], in view of the earlier published work, the validity of the patent is doubtful. Additional publications of essentially the same mechanism can be found in Hertz and Hughes [1993] and Peruzzini et al. [1995]

The mechanism consists of three actuated arms attached to a triangular base through equally spaced revolute joints. Three passive arms are attached to an upper platform in the same way as the driven arms are attached to the triangular base. Each of the three base arms is connected to its respective opposing passive arm via a spherical joint. For the particular case where the passive arms and upper platform are identical to the driven arms and base, the three spherical joints define the homokinetic plane of the true constant velocity joint formed between the base and the upper platform of the complete mechanism. The mechanism depicted in figure 1.8 has three degrees of freedom, two rotational and one translational.

1.5.3 Two axis parallel mechanism

It is possible to reduce the three dof mechanism to only two rotational degrees of freedom by introducing another linkage running from the centre of the base to the centre of the upper platform as in figure 1.9. In this configuration only two axes need be driven, therefore simplifying the control of the mechanism.

It is the two mechanisms shown in figures 1.8 and 1.9 and similar derivatives upon which this project is based.

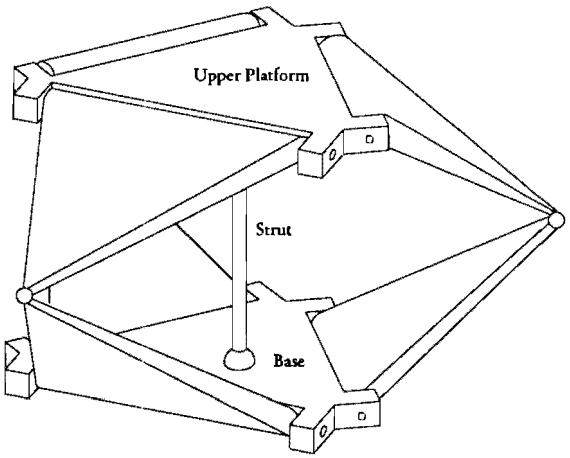


Figure 1.9 Two axis parallel link mechanism.

1.6 Outline of material covered in the remainder of the thesis

In Chapter 2, some two and three dof spatial mechanisms are briefly examined from the point of view of being candidate mechanisms for beam-aiming operations. The chapter finishes with some preliminary observations of the two and three dof mechanisms chosen for this project.

The design of the robotic mechanism is discussed in Chapter 3. The design preceded the analyses developed in later chapters and so only rudimentary calculations using estimated loading scenarios could be carried out. The control of the robot is discussed in Chapter 4 with brief descriptions of the hardware and the layout of the control program. Some comments on observations of the actual moving robot are made.

The geometry of the two and three dof mechanisms is analysed in Chapter 5. Closed form solutions for the direct and inverse kinematics are developed for both mechanisms. A numerical example is computed to demonstrate the sixteen possible solutions for the direct kinematic problem.

A static analysis using a more traditional approach of vector statics is performed in Chapter 6. The results of this analysis compare favourably with experimental results obtained by direct measurement of the forces using spring balances.

In Chapter 7, the kinematics of an idealised orbiting object are derived. This is an idealised model of a satellite path for use in the inverse dynamic formulations which are developed in the chapters which follow. The kinematics of large base motions simulating a moving ship are also developed.

The dynamics of open chains containing rotational and prismatic joints are derived in Chapter 8. This chapter is largely based on the Newton-Euler approach developed by Hughes [1988]. There is some departure in notation, but a reader familiar with this work may wish to skip this chapter. The dynamics of multiple looped closed chained mechanisms are derived in Chapter 9. The dynamics of a prismatically actuated two dof mechanism is used as an example to illustrate the formulation of the dynamics.

In Chapter 10, the dynamics of five similar mechanisms are compared using the dynamic formulation developed in the preceding chapters. Counterbalancing of the mechanisms is introduced to illustrate the significant reduction in actuator requirements for a specified movement. The relative merits of using each of these mechanisms for beam-aiming operations in discussed. A maritime simulation is carried out to show the effect of a moving ship on the actuator requirements and the link reaction forces.

Finally the conclusions are drawn in Chapter 11 together with recommendations for future work.

Chapter 2

A review of some two and three dof spatial mechanisms

2.1 Introduction

Although the project brief was to design and build only the two antenna aiming robots it would have been rather narrow to focus the study on these mechanisms only. Hence a number of other similar parallel mechanisms were considered, although not in great detail. The primary constraints were that each mechanism should be able to point anywhere in the upper hemisphere while encountering no singular positions. There may be, and probably are, many other mechanisms that would be suitable for consideration. This is by no means an exhaustive list, in fact no formal method of synthesis has been discovered and the mechanisms shown in the following pages were found in the literature.

2.2 A spherical three dof parallel manipulator

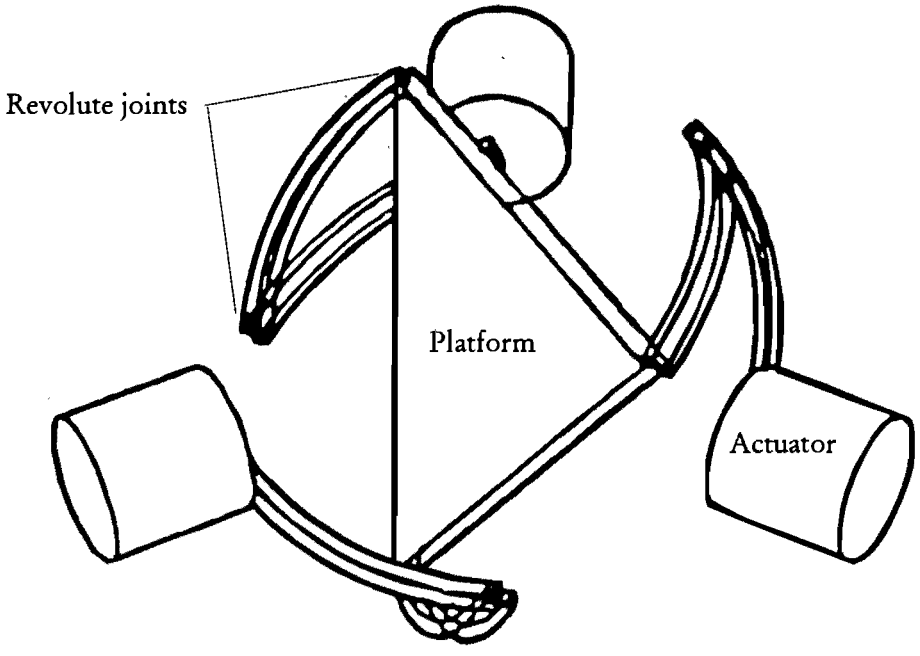


Figure 2.1 Spherical three dof parallel manipulator.

The manipulator shown in figure 2.1 has been studied by Gosselin and Angeles [1989]. Although it is often claimed to have three degrees of freedom, this is only true for a special case. Gosselin and Angeles state that all spherical three dof manipulators with revolute joints require their axes of rotation of all the joints to intersect at a common point called the

geometric center of the manipulator. If this requirement is not met the mechanism will be over constrained i.e. a mobility of less than 0 (see equation 2.1). The fact that this mechanism is able to exhibit a mobility of 3 for this particular configuration is explained by Phillips [1984] and this situation is referred to as a *special mobilty*. The main problem with such mechanisms is that without releasing some freedoms, i.e. introduce some two dof cylindrical joints for the one dof revolute joints, when the mechanism is actually built it may and probably will bind. Only if there is enough elasticity and/or clearance in the joints will the mechanism move freely. Unfortunately for most robot applications, including beam-aiming applications, it is not desirable to have sloppy joints or elastic links since the positioning of the platform is adversely affected. Also when this mechanism is considered for an antenna mount, the interference or fouling of an antenna with the links and actuators is a major drawback.

The mobility of the mechanism is given by Kutzbach's criterion

$$\begin{aligned} \text{Mobility} &= 6(n-g-1)+\sum f \\ &= 6(8-9-1)+9 \\ &= -3 \quad \text{i.e. over constrained.} \end{aligned} \tag{2.1}$$

where n = number of links
g = number of joints
and $\sum f$ = sum of the freedoms

2.3 A three dof parallel manipulator with collinear actuators

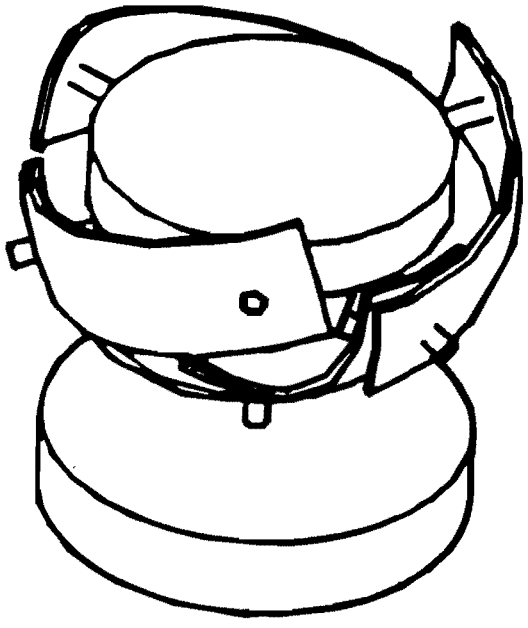


Figure 2.2 Three dof parallel manipulator with collinear actuators.

The manipulator shown in figure 2.2 has been studied by Gosselin and Lavoie [1991]. It has a special mobility of 3, but suffers from the problems outlined in section 2.2 since it has an apparent mobility of -3.

2.4 A three dof parallel manipulator with coplanar actuators

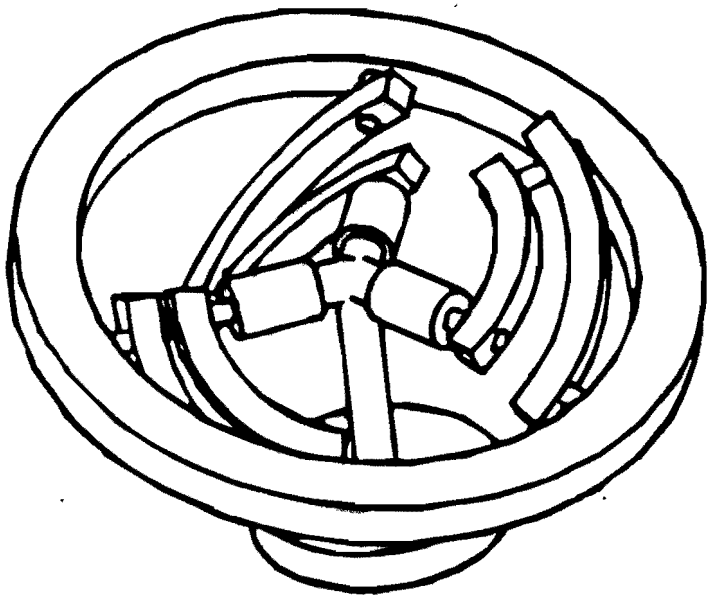


Figure 2.3 Three dof parallel manipulator with coplanar actuators.

The manipulator shown in figure 2.3 has been studied by Gosselin and Lavoie [1991]. Again it has a special mobility of 3, but also suffers from the problems outlined in section 2.2 since it has a mobility of -3.

2.5 A three dof shoulder module

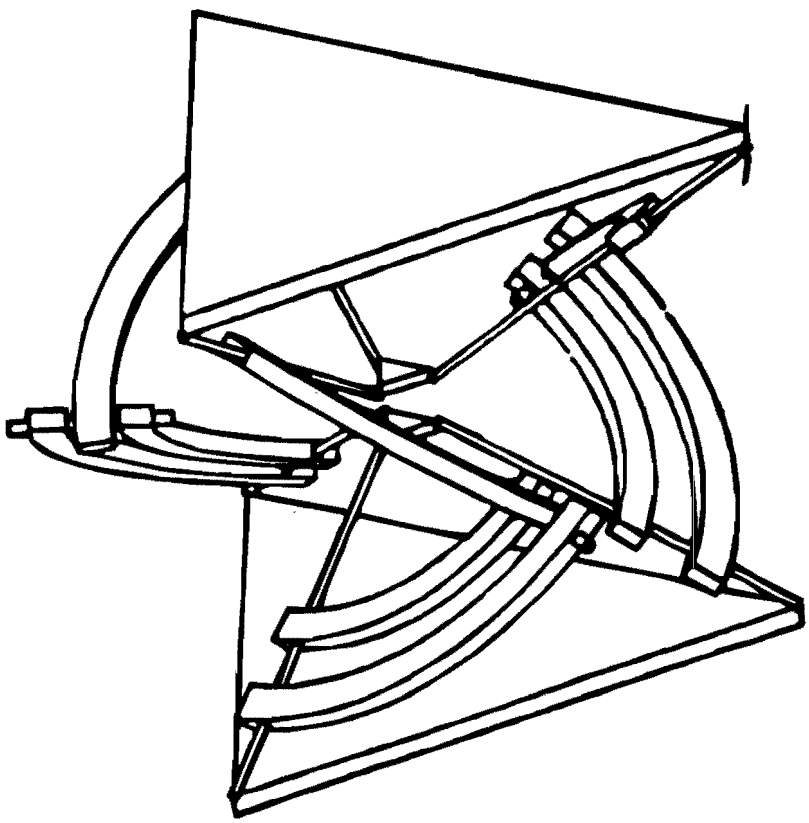


Figure 2.4 Three dof shoulder module.

The manipulator shown in figure 2.4 has been studied by Gosselin and Lavoie [1991] and Alizade et al [1994]. Again it has a special mobility of 3, but also suffers from the problems outlined in section 2.2 since it has a mobility of -3.

2.6 A three dof triple arm manipulator

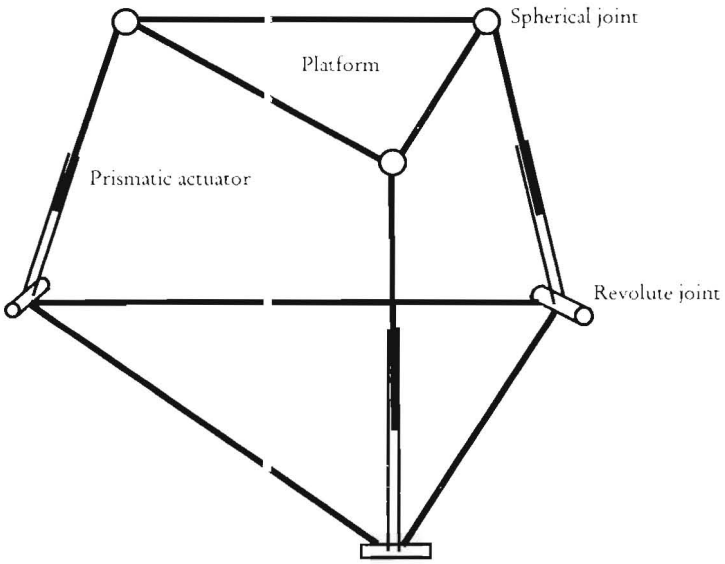


Figure 2.5 Three dof triple arm manipulator.

The manipulator shown in figure 2.5 was presented by Lee and Shah [1988]. It has two degrees of orientation freedom and one degree of translatory freedom. Unlike the mechanisms shown previously it is not over constrained but it does have a singularity in the upper hemispherical pointing direction. This singularity will be experienced whenever a spherical joint passes through the plane formed by a revolute joint and the two other spherical joints.

2.7 The UNITRU coupling

A brass model of a UNITRU coupling from South-Western Instruments (California) is shown in figure 2.6. It has a special mobility of 2, but suffers from the problems outlined in section 2.2 since it has a mobility of 0.

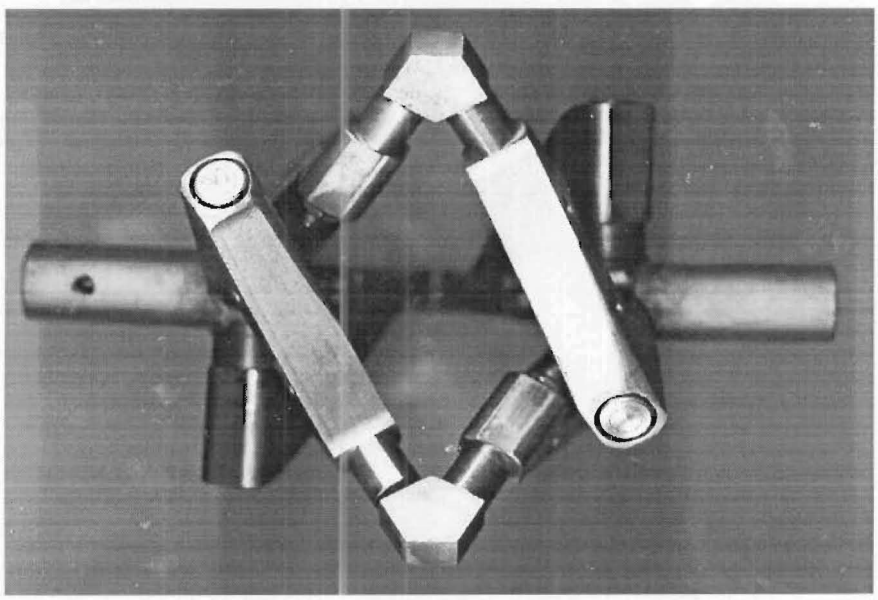


Figure 2.6 A model of a UNITRU coupling.

2.8 Some observations on the chosen two and three dof mechanisms

Small cardboard and plastic models of the two and three dof mechanisms shown in figures 1.8 and 1.9 were initially constructed so that the movements of the mechanisms could easily be studied. Note that for the three dof mechanism, if a larger arm length to base/platform ratio is used as shown in figure 2.7, then more extreme elevation angles can be realised before a singular position is reached. The disadvantage in having a high ratio is higher actuator torques and increased flexibility.

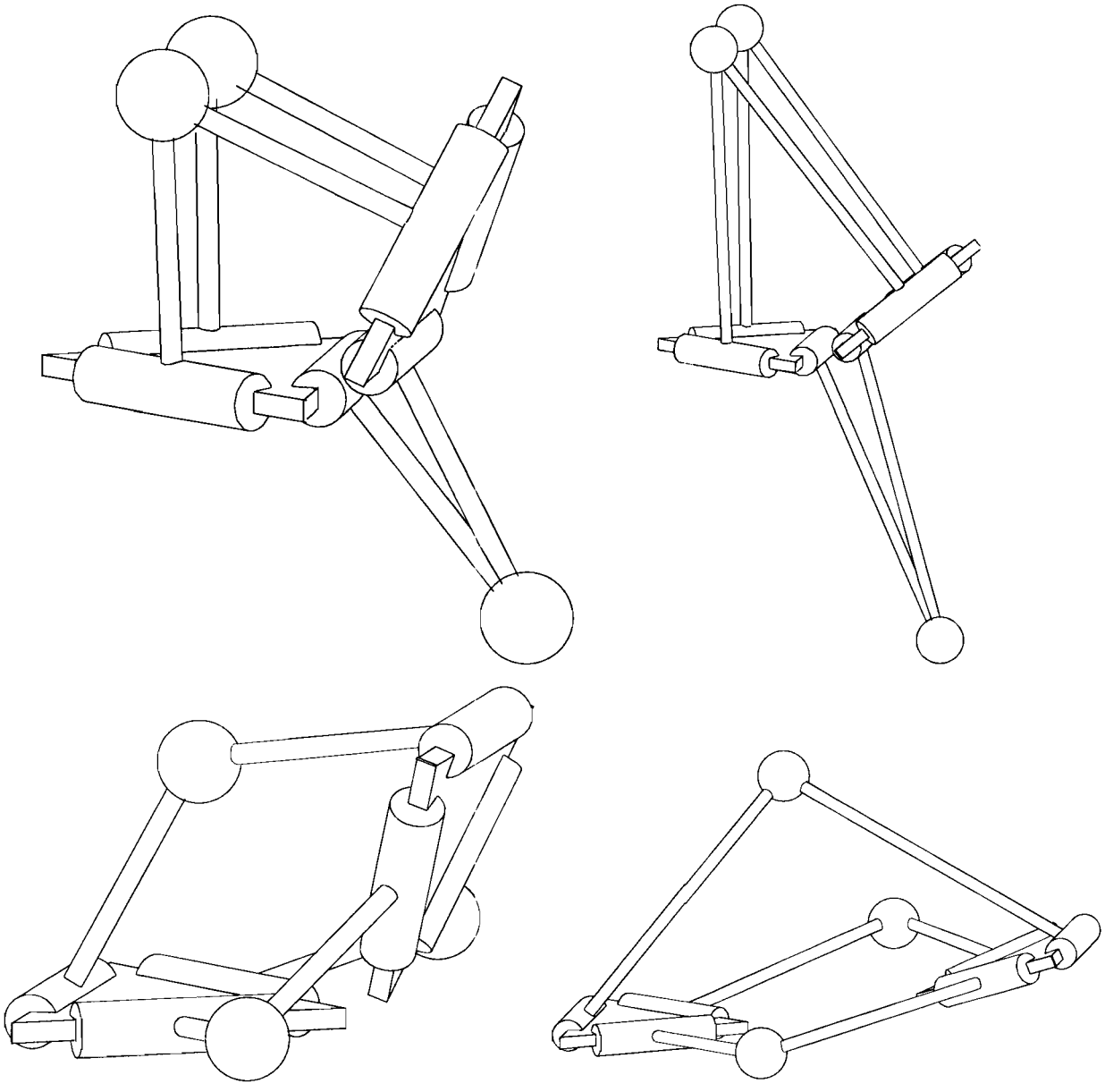


Figure 2.7 Three dof mechanism showing the increased extreme elevation pointing angle able to be achieved with a larger arm length to base/platform ratio.

The two dof mechanism introduced another variable parameter, the strut length. Upon varying these three parameters, no matter what ratio was used, at best the two dof mechanism was restricted to elevation angles from just below the horizon and above. It was noticeable that one strut length did not give the best range in elevation angles for all azimuthal angles. Depending on the azimuthal angle, a longer or shorter strut length would allow the mechanism to achieve lower elevation angles. Thus it was obvious a compromise would be needed, and ideally an

optimum strut length would need to be found which would allow the best coverage in all azimuthal directions.

As suggested by K. Hunt in a private communication, greater stiffness may be achieved by offsetting the arms as shown in figure 2.8. This configuration also has the advantage that there is less interference between the arms and the antenna when pointing at a low elevation. Unfortunately this was not a viable option in this project as the driving mechanism had to be compatible with the DELTA robot developed at the University of Canterbury by Dunlop et al. [1992].

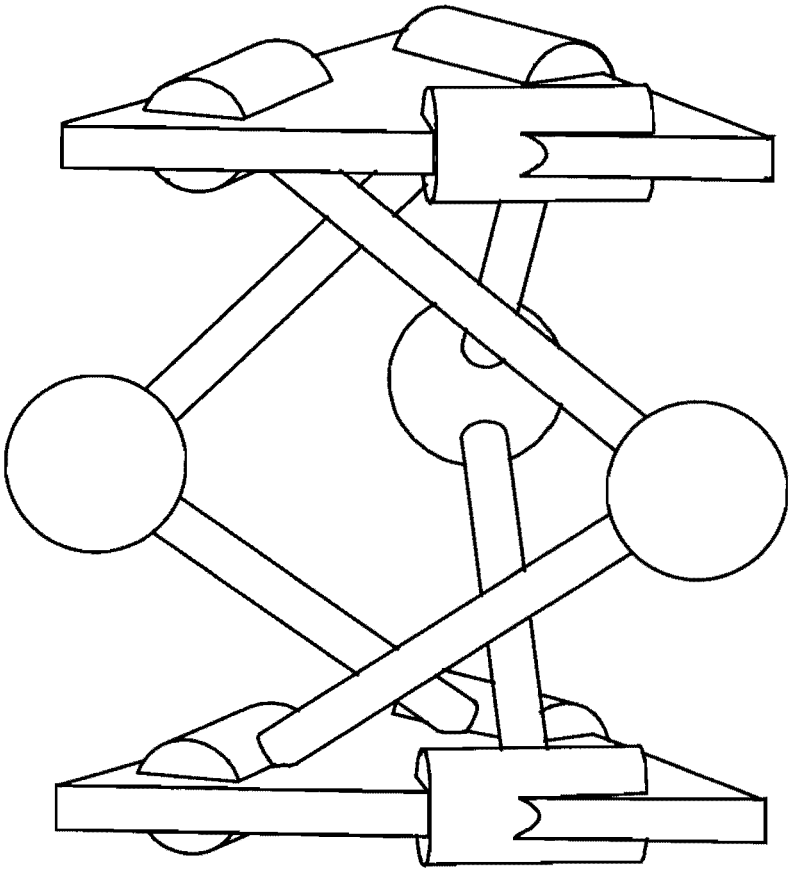


Figure 2.8 Three dof mechanism with offset arms.

In fact, no formal optimisation of the parameters was carried out in this project, although some manual adjustment of the parameters was done to obtain a rough idea of the optimum strut length for a given arm length to base/platform ratio. Eventually an arm length to base/platform ratio of 1 was used as this enabled both mechanisms to cover the upper hemisphere without encountering a singularity.

Chapter 3

Design of the robot

3.1 Introduction

The main objective for the design of the robot was to construct a robot that would work sufficiently well to prove the concept of using these particular mechanisms for beam-aiming applications. Little rigorous force/stress analysis could be carried out in the design stage since only the forward kinematics had been investigated at this point. Fairly crude estimates of stresses experienced by the links and joints were made where it was thought necessary but a large safety margin was included in the design. The robot was essentially designed for rigidity and was hence fairly bulky; thus failure was not expected even if the estimates were wildly incorrect. This design procedure led to what is probably a fairly over-designed robot with much excess weight that could be trimmed to achieve a more optimal design. Engineering drawings of all components discussed in this chapter can be found in Appendix A.

3.2 Design of the transmission system

3.2.1 Introduction

The driven arms need to be precisely positioned, therefore there must be minimal flexibility and backlash in the transmission system. The transmission options considered for this robot are listed below with a brief discussion of their relative merits.

1) Toothed belt transmissions were considered to have too much flexibility and not able to provide an adequate single stage reduction ratio.

2) Pre-loaded worm gears have an advantage of high reduction ratios and no backlash, but the pre-loading has the disadvantage in that it significantly lowers the transmission efficiency.

3) Cyclo drives are a combination of an internal planetary gear mechanism with a cycloidal tooth profile. They are offered with reduction ratios from 1:6 to 1:119 for single stage units, and have an efficiency of over 90%. A special FA and FR series were conceived for precision control applications such as for robots but these were not readily available in NZ. The FA and FR series drives are ideal for robotic applications, having negligible flexibility and no backlash, but high cost and local availability precluded their use.

4) Harmonic drives consist of only three components, a wave generator, a flexspline and a circular spline. They are available with reduction ratios between 1:50 and 1:320. A low weight and compact transmission construction can be produced since they are able to produce high transmission ratios in a single step unit together with a co-axial construction of input to output. This, coupled with an efficiency of between 75-90%, moderate to high stiffness, and backlash of less than 3 arc seconds make the harmonic drive ideal for robotic applications.

3.2.2 The harmonic drive gear box in more detail

The designer has essentially three options in drive configuration as shown in figure 3.1.

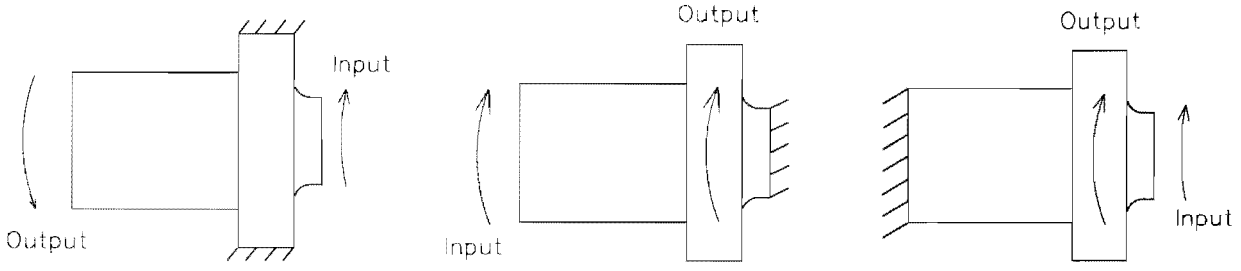


Figure 3.1 Possible configurations for a harmonic drive.

The third configuration is suited for this application since it is very compact, allows for easy attachment of external arms and provides the lowest transmission flexibility. This can be compared to transmission systems (c.f. figure 3.2) where a linear arrangement of a motor, a gear box and the driven component is common. This configuration allows for relatively high flexibility, i.e. torsional twisting movements in the shafts and couplings. Thus unwanted positional errors are introduced into the system.

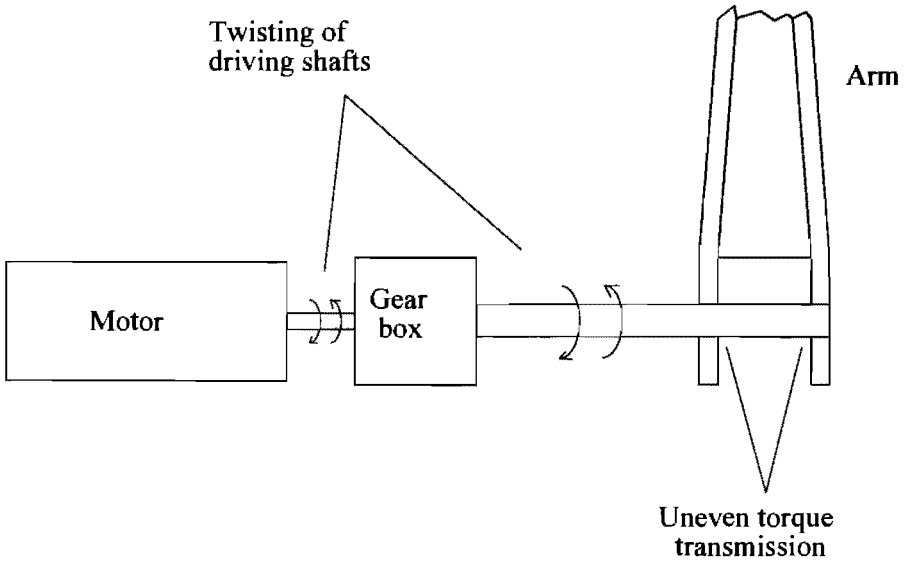


Figure 3.2 Conventional transmission system.

The driven arm can be attached to the harmonic drive via a large diameter 'torque' tube shown in figure 3.3. This configuration is very stiff and ensures that equal torque is able to be transmitted to both sides of the arm, thus preventing unwanted twisting along the longitudinal axis of the arm. Since a large diameter tube is needed to encase the harmonic drive, deep arms can be easily attached and these are ideal for transmitting high loads with minimal deflection.

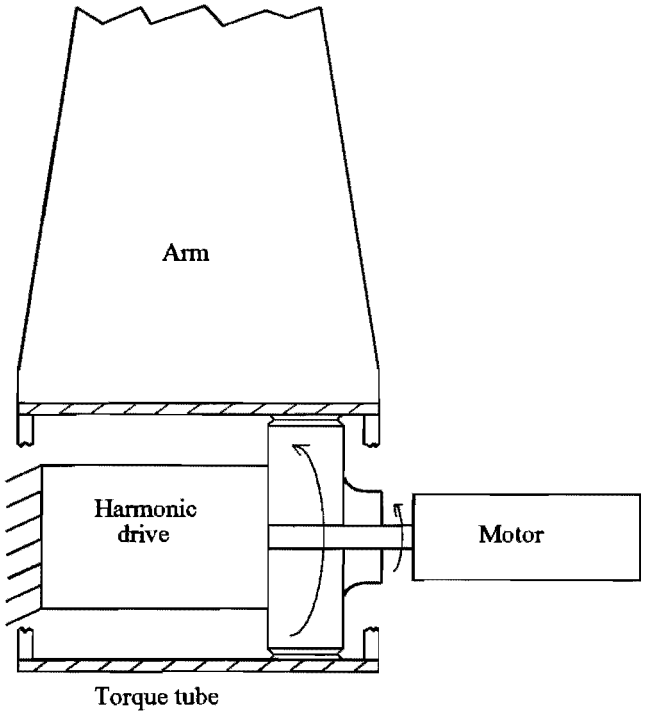


Figure 3.3 Harmonic drive with attached arm.

The are a number of different ranges of harmonic drives offered by the manufacturer (Harmonic Drive Technologies), all having their own particular advantages. The HDUC type was found to be best suited for this robot due to its relatively high torque handling capacity. The other drives offered, such as the pancake drive and the compact drive, although smaller and lighter, had significantly lower torque handling capacities.

At this early design stage it was apparent from looking at the overall geometry of the robot and taking into account the size of the motors, that the arms would be about 1m long; in fact they ended up being 0.8416m long. It was assumed that each arm would lift approximately 100kg. Therefore the drive would have to be rated at about 1000Nm. Ideally the 65 series harmonic drives with a ratio of 78:1 would have been the most desirable, as they have a continuous torque capacity of about 1000 Nm. They would also be well matched to the motors if the motors were driven at their full 13 Nm torque output. Due to cost and availability considerations, a set of three 50 series drives with continuous torque capacities of about 530 Nm and gear ratios of 80:1 were purchased. It is possible to overrate these harmonic drives at the expense of their operating life with an absolute maximum recommended torque output of about 1200Nm so the desired 100kg lifting load could be attained if absolutely necessary.

The computer-aided design package MicroStation was extensively used in the detailed design of the gearbox housing. Its use was essential in ensuring that the components of the gearbox fitted together in the desired manner. A part cut away 3D computer-rendered view of the gear box is shown in figure 3.4 and a 2D cross sectional view shown in figure 3.5.

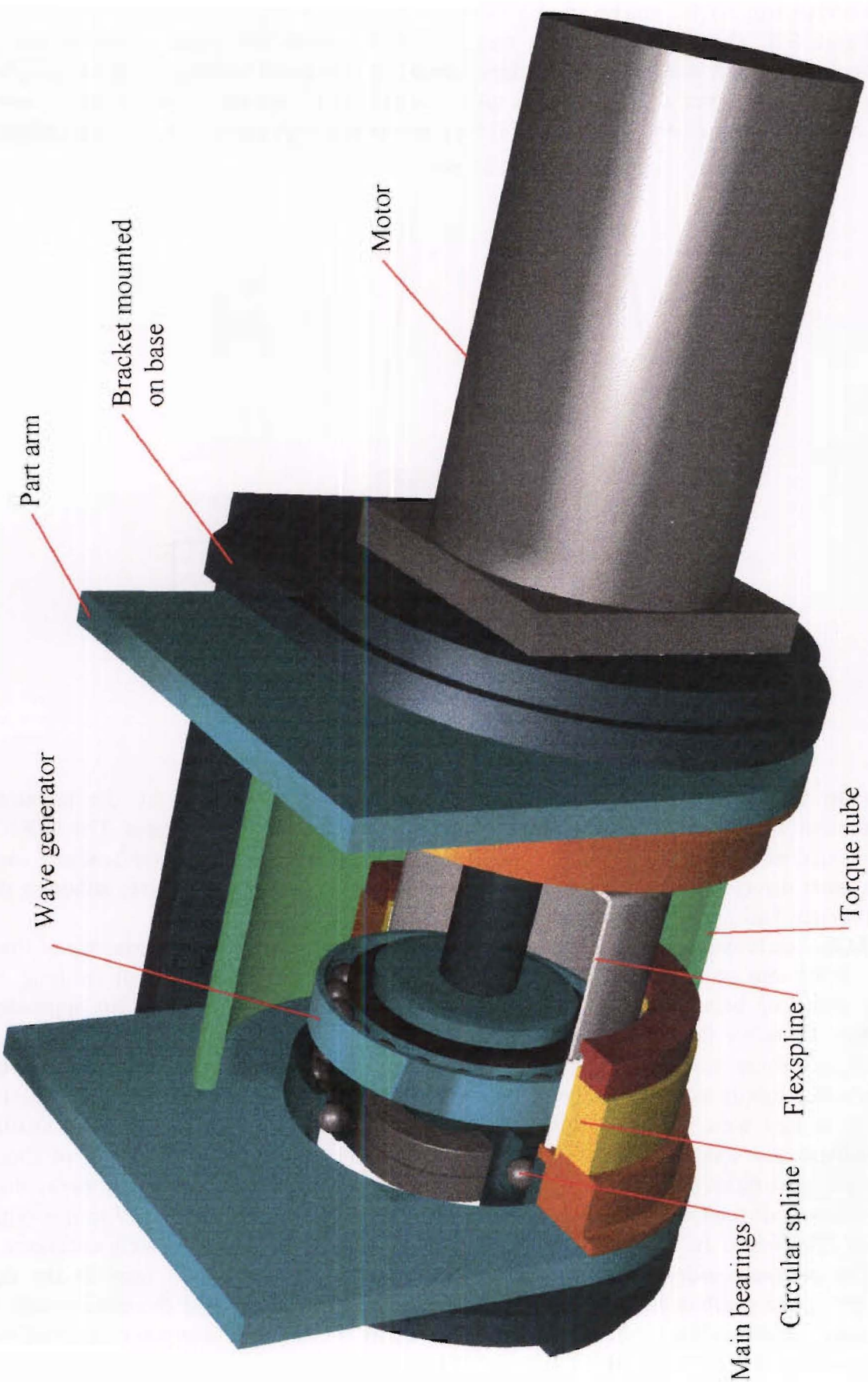


Figure 3.4 3D computer-rendered view of the harmonic drive.

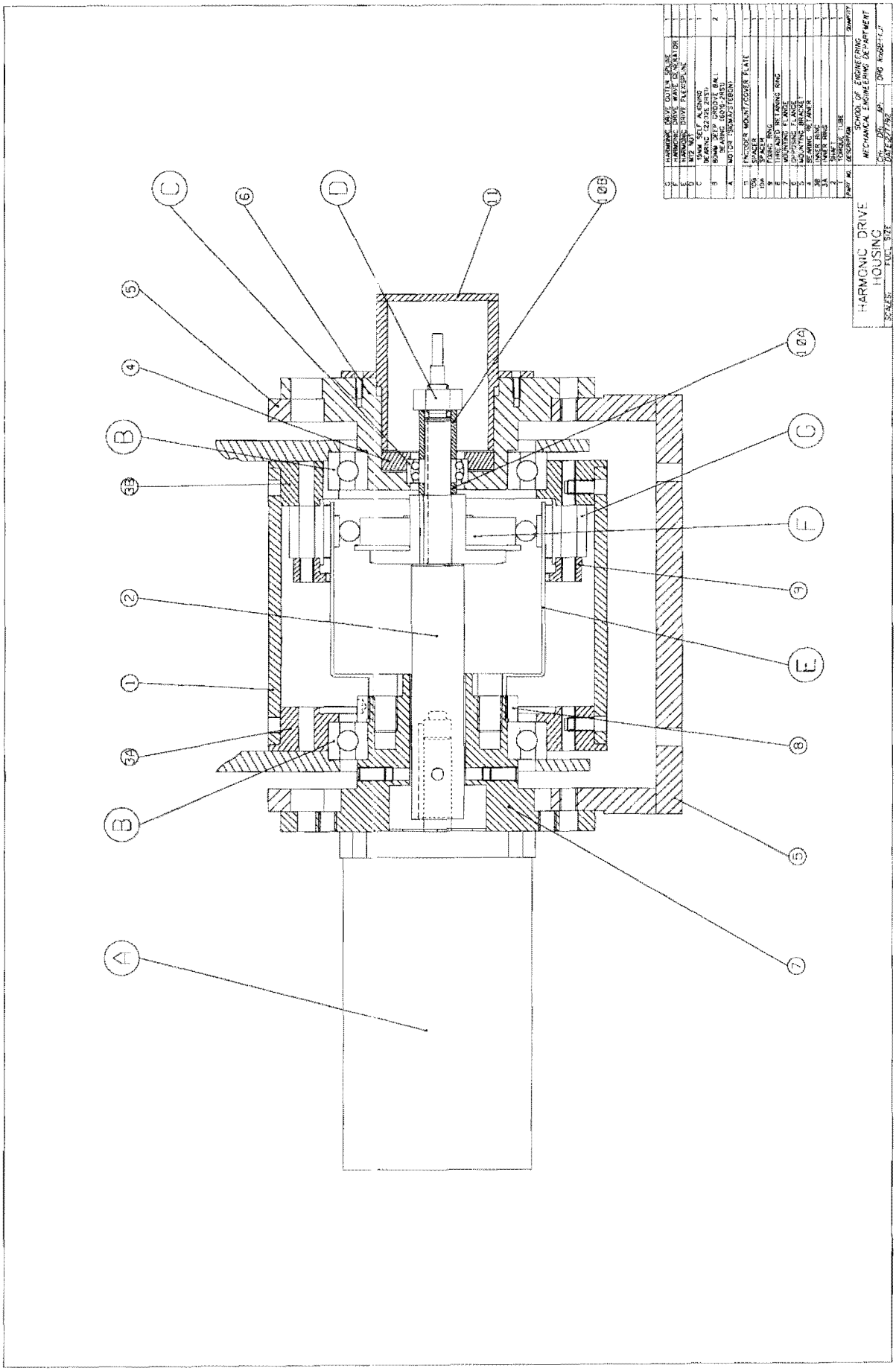


Figure 3.5 2D Cross sectional view of the harmonic drive.

The two most important tolerances required to be met were the axial and radial positions of the flexispline relative to the circular spline. Both tolerances were very tight and so all components that affected the position of these components had to be machined so that their combined tolerances (tolerance stack) were not greater than the maximum allowable position and geometric tolerance of the splines. There is a small amount of play in the main deep groove ball bearings which will affect the absolute positioning of the arm but the effect of the bearing play is insignificant when compared to the errors due to the machining tolerances of the rest of the robot, and so for this reason deep groove bearings were chosen. To eliminate the bearing float completely preloaded angular contact bearings could be used, but at a cost of added complexity due to the requirement of end float adjustment nuts.

Not shown in figure 3.5 are shims that were placed between the main bearings and part 6; these ensured that the outer housing could not slide axially and was therefore always located relative to the left hand main bearing. Rotary encoders were not used in this project but provision was made for the attachment of a rotary encoder on the high speed side of the gearbox (c.f. figure 3.5, part 11).

The gear box was designed so that the arms could slide into position around the main bearings and then be bolted on to the gearbox. This ensures that the arm is accurately positioned about its axis of revolution. The completed gear box weighed 37.05kg and combined with the step motor (10.54kg) weighed 47.59kg.

3.3 Design of the base

The base was designed to provide a very rigid mounting structure for the accurate attachment of the gear box mounting brackets, attaching the supporting legs, and also provides a place to attach the ball joint joined to the centre strut. To make the robot as small as possible, the base platform was designed to be as small as practicable, as this in turn determines the length of the arms. The limiting factor in the scaling of the base platform was the attachment of the supporting legs so that they would not interfere with the motors.

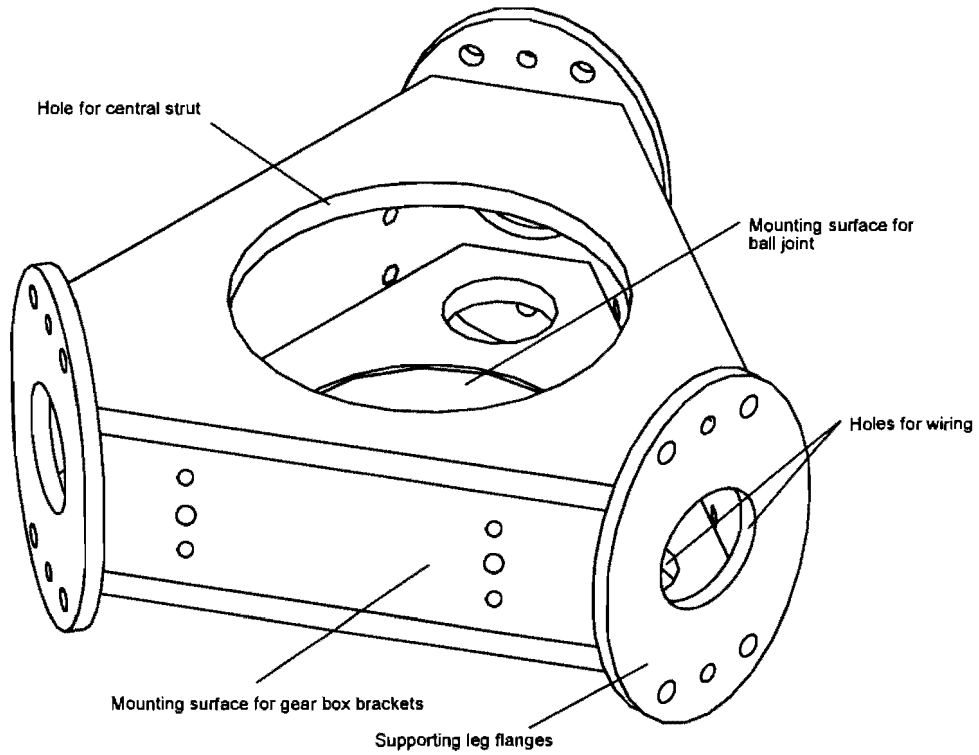


Figure 3.6 Base unit showing external attachment points.

3.3.1 Design of the base in more detail

The base platform shown in figure 3.6 was designed as a single welded steel component rather than a series of bolted sides; thus no relative movements of the mounting surfaces could occur once the structure had been assembled. The entire structure was stress relieved after welding so that it would not deform as it was being machined. It was made with accurately milled surfaces to give correct alignment of the gear boxes, thereby ensuring that the driven arms would rotate about well-aligned axes. The completed base weighed 32.31kg.

A central mounting surface was provided for a ball joint to facilitate the attachment of a central strut, so that the robot could be run in the two dof configuration. The mounting surface was machined so that the centre of the ball lies in the plane made by the rotational axes of the gear boxes. A large hole was cut from the top of the base unit to allow for the central strut.

3.4 Design of the support legs

The base was designed so that it could be supported from the underneath by a column support as shown in figure 3.7 or from three side attachment positions as shown in figure 3.8. A spigotted mounting surface was machined on the under side of the base unit to allow for accurate attachment of a single column support leg. This is an optional support leg that is not compatible with the Delta robot.

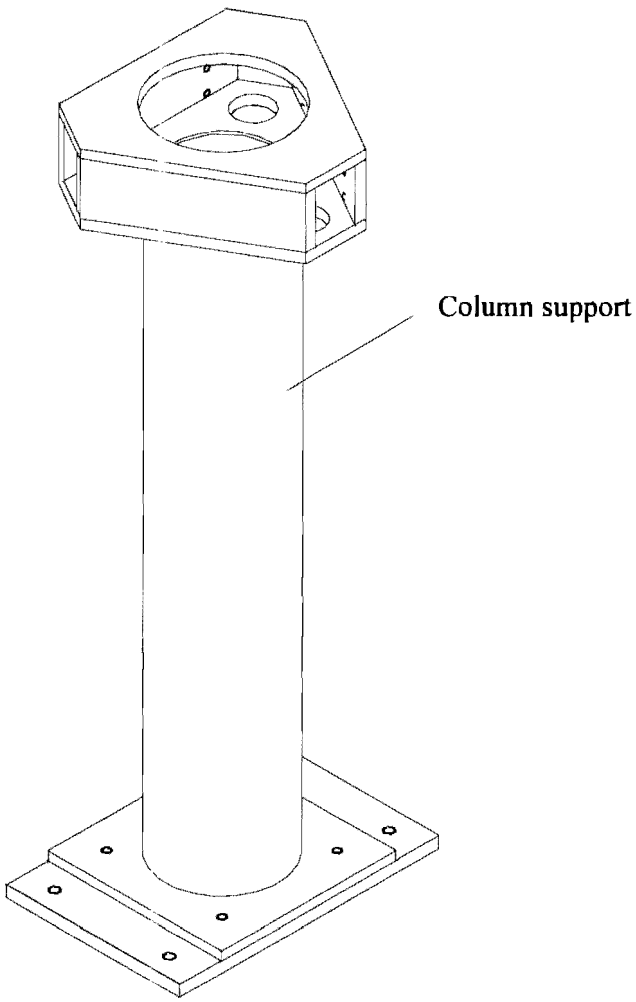


Figure 3.7 Base unit with optional column support.

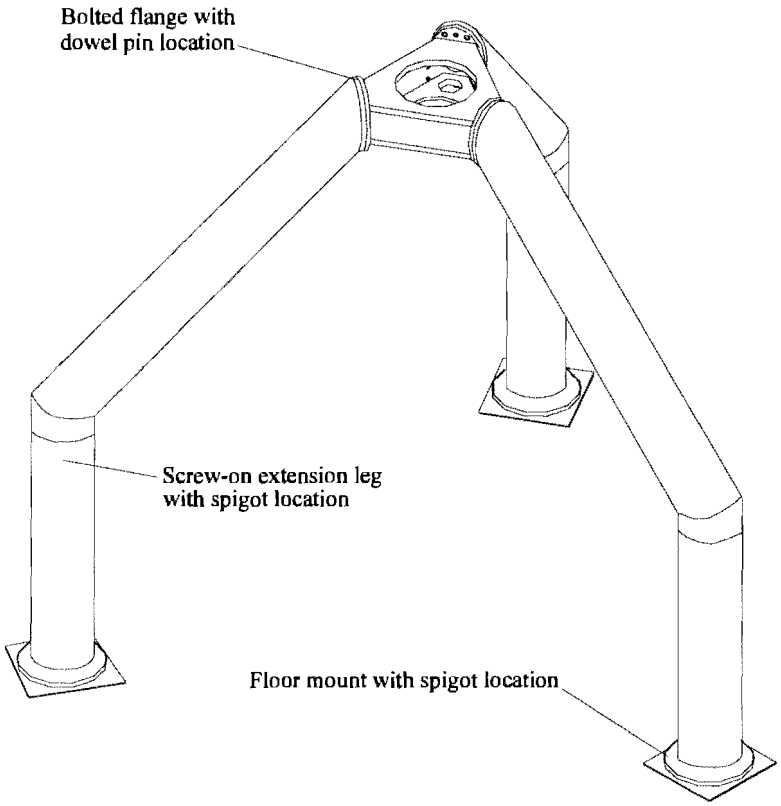


Figure 3.8 Base unit with three supporting legs.

3.4.1 Design of the support legs in more detail

Extruded aluminium tube was used as for the support legs; this had the advantage of being relatively cheap and needing minimal fabrication effort. The tube was flanged at each end, with one end having a simple bolted type arrangement to the base unit and the other a screw type arrangement for easy attachment of the extension legs. Provision for wiring was made by including a hole in the flange that is attached to the base unit.

Screw-on extensions to be attached to the bottom of the legs were made so that the height of the base could easily be altered by attaching different length extensions. This requirement was particular to the Delta robot only, so that its walking application could be realised. The extensions were given flanged feet so that they could be easily bolted to level floor mounting blocks if required.

The total weight of the robot including pay load was estimated at approximately 300kg. It was assumed that the weight will not always be evenly distributed among the three legs as the centre of mass will move as the robot moves into different positions. A design load of 175kg for each support leg was used to take into account the increased loading on any particular leg as the centre of mass moves towards that leg.

The support legs may experience two different loading scenarios in normal operation as shown in figure 3.9. The leg can either be free or bolted to the ground. The highest stress will occur when the leg is unconstrained and so this is used as the design case. For the sake of simplicity, the leg is modelled as a cantilevered beam as shown in figure 3.10. Although it is not strictly correct that the leg experiences only bending stresses (there will be some compressive stresses due to the angle the leg makes with the base), this loading model is adequate since it is conservative.

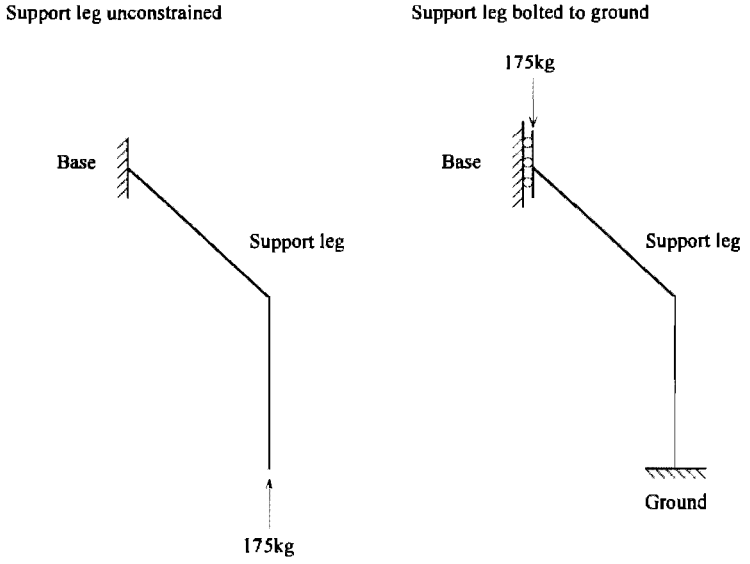


Figure 3.9 Loading scenarios for the support legs.

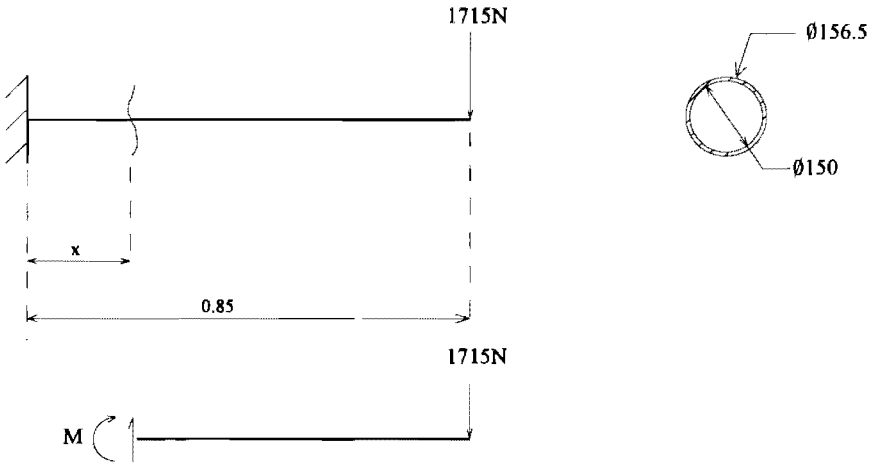


Figure 3.10 Loading model for the supporting legs.

Summing the moments about M we get

$$\sum \mathrm{M} = 0: \quad \mathrm{M} + 1715(0.85 - x) = 0$$
$$\mathrm{M} = -1715(0.85 - x).$$

The largest moment M = -1460Nm occurs when $x = 0$. The maximum bending stress occurring in the leg is then given by

$$\sigma_b = \frac{My}{I} = \frac{-1460 \times 0.075}{7.35 \times 10^{-5}} \qquad I = \tfrac{\pi}{4}(d_o^4 - d_i^4)$$
$$= 1.5\,Mpa.$$

The maximum bending stress is well below the yield strength (70MPa) of the aluminium. There is some uncertainty about the effect of the flange welds on the reduction in strength, but the legs should still be operating in a safe stress region.

The deflection of the legs due to the 175kg load can be found by solving the differential equation for the deflection of a uniform beam

$$\frac{d^2w}{dx^2} = -\frac{M}{EI}$$

to get

$$w = \frac{Pl^2x}{2EI} - \frac{Pl^3}{6EI}$$

$$= \frac{1715 \times 0.85^2 \times 0.85}{2 \times 70 \times 10^9 \times 7.35 \times 10^{-5}} - \frac{1715 \times 0.85^3}{6 \times 70 \times 10^9 \times 7.35 \times 10^{-5}}$$

$$= 0.07mm.$$

3.5 Design of the lower arms

The lower arms were designed to have high rigidity and to be readily attachable to the gear box drive system. It was also necessary that they be able to cater for the Delta robot. They were fabricated from aluminium plate that was bolted together to form a tapered box section type structure. Due to limitations in the travel of the mill most of the machining had to be carried out before assembly. If a mill with larger travel had been available, the machining would have been carried out post assembly so that better tolerances could be achieved. The length of the arms was set by the size of the base in order to preserve a one to one geometric ratio between the perpendicular length of the base and the arm length. The weight of each arm is 8.81 kg.

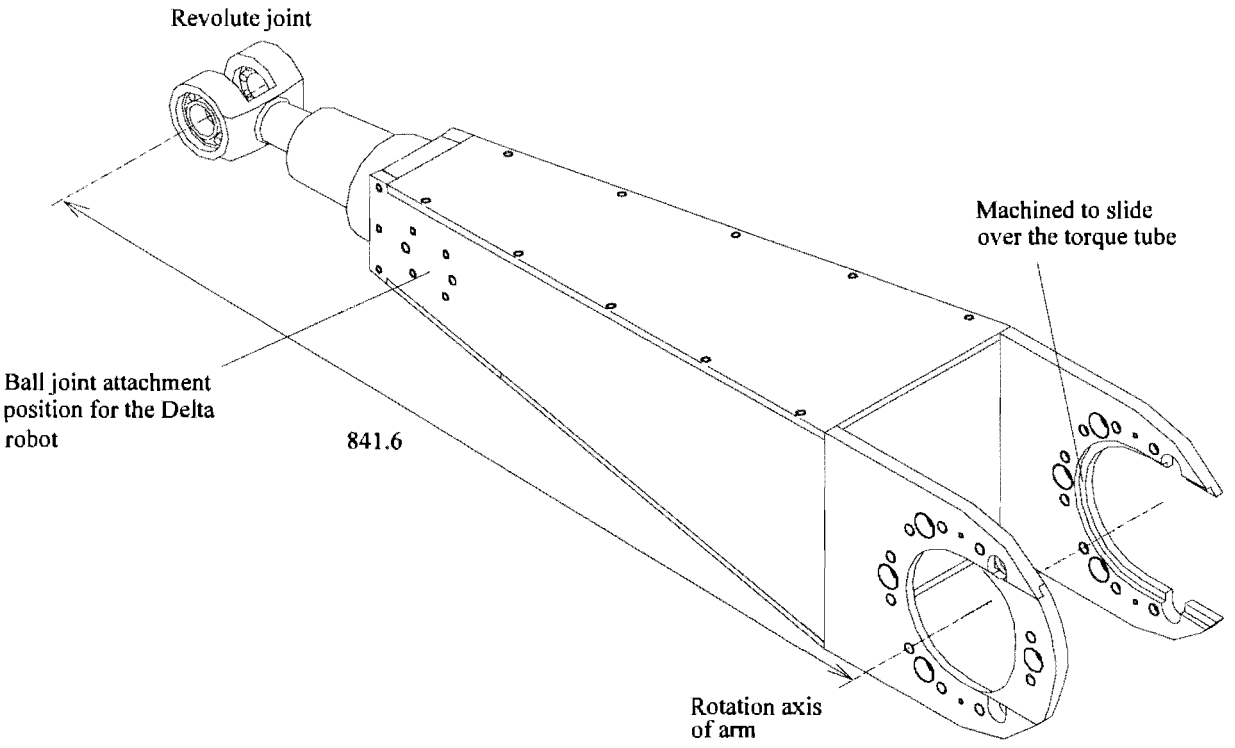


Figure 3.11 Lower arm with revolute joint attached.

3.5.1 Design of the lower arms in more detail

The arms were designed so that they could be easily attached to the harmonic drive gear box. This was achieved by machining an accurate slot in the inner side walls of the arms as shown in figure 3.11 so that they could slide onto the gear boxes and be accurately located. Side walls of 12mm aluminium were used with a 5mm deep slot leaving a 7mm thick flange. Provision was made for mounting the revolute joints for the spherical robots or two ball joints for the Delta robot. The ball joint attachment points are 600mm from the rotation axis of the arm, and the revolute joint is 841.6mm from the rotation axis of the arm. Since the lower arms and passive arms need to close together in some positions and roll around each other, it was necessary to taper the arms in both directions; this was achieved by bending the 12mm side walls at a predetermined angle. A tapered 6mm top and bottom were added to form a box section structure to increase the rigidity.

The normal arm loading possibilities for the spherical robots are a combination of:
a) bending about the rotational axis of the arm,
b) bending about an axis perpendicular to the rotational axis of the arm,
c) compression along the longitudinal axis of the arm and
d) tension along the longitudinal axis of the arm.

The worst loading scenario experienced by the arms was loading possibility (a) shown in figure 3.12. Since the lifting force of the arms is based upon the maximum torque output of the gear boxes, this gave a maximum lifting force of about 1425N or 145 kg.

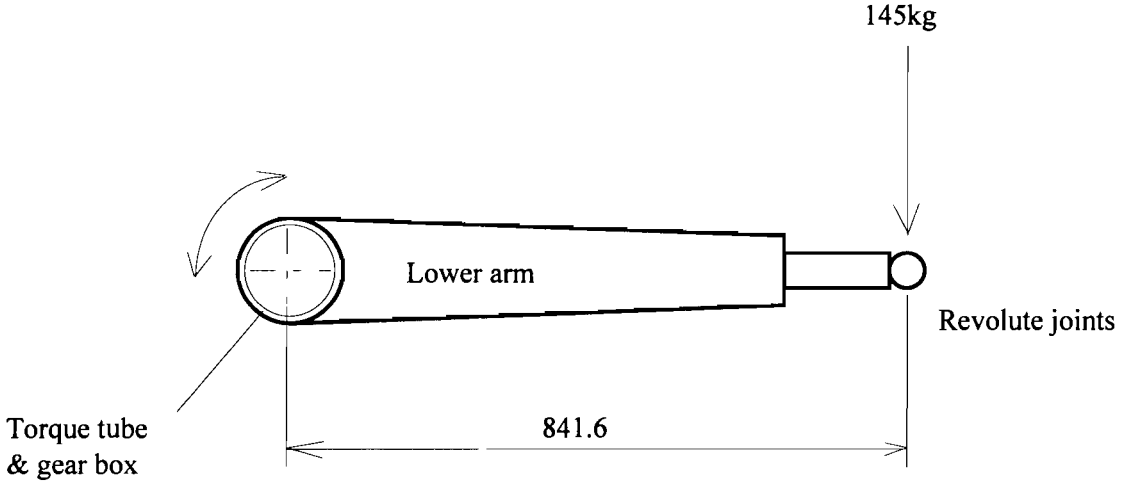


Figure 3.12. Maximum arm loading.

The deflection of the arms is important since an error in the pointing direction of the antenna will occur unless the kinematics includes the stiffness of the links. Although no maximum deflection could be specified at this stage of the design process it was thought that the maximum deflection should be less than 1mm.

The loading model for the lower arms is shown in figure 3.13 below.

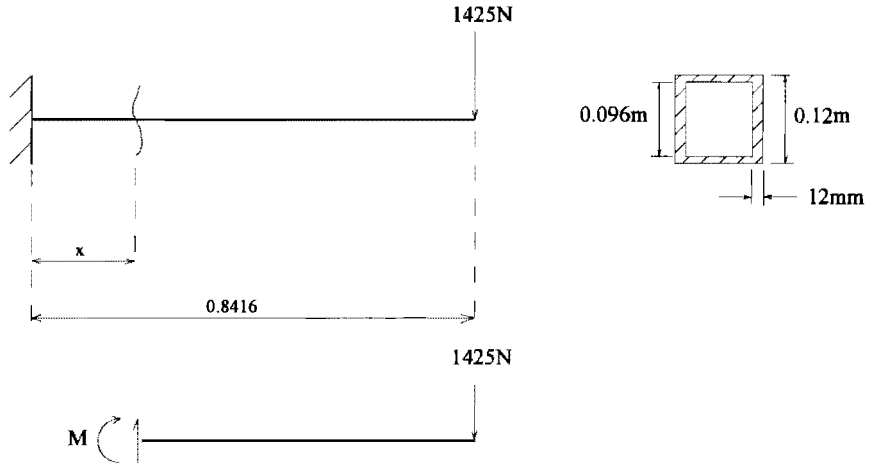


Figure 3.13 Loading model for the lower arms.

Summing the moments about M we get

$$\sum \mathrm{M} = 0: \quad \mathrm{M} + 1425(0.8416 - x) = 0$$
$$\mathrm{M} = -1425(0.8416 - x).$$

The largest moment M = -1200Nm occurs when x = 0. Assuming that the arm is uniform along its length, i.e. does not taper, the deflection of the arm due to the 145kg is then

$$w = \frac{Pl^2x}{2EI} - \frac{Pl^3}{6EI} \qquad I = \frac{a^4 - b^4}{12}$$
$$= \frac{1425 \times 0.8416^2 \times 0.8416}{2 \times 70 \times 10^9 \times 1.02 \times 10^{-5}} - \frac{1425 \times 0.8416^3}{6 \times 70 \times 10^9 \times 1.02 \times 10^{-5}}$$

$$= 0.4mm.$$

3.6 Design of the upper arms

The upper arms were designed similarly to the lower arms. The arms were machined with an accurate slot in the inner side walls as shown in figure 3.14 so that they could slide on to the platform tubes and be accurately located. 10mm aluminium was used for the side walls with a 5mm slot and leaving a 5mm thick flange. Provision was made for mounting the revolute joints using a spigot-type location to ensure accurate positioning. The rotation axis of the revolute joint was 841.6mm from the rotation axis of the arm. The 10mm side walls were bent at a predetermined angle and a tapered 6mm top and bottom was added to form a box section like structure to increase rigidity. The weight of each arm is 8.15kg.

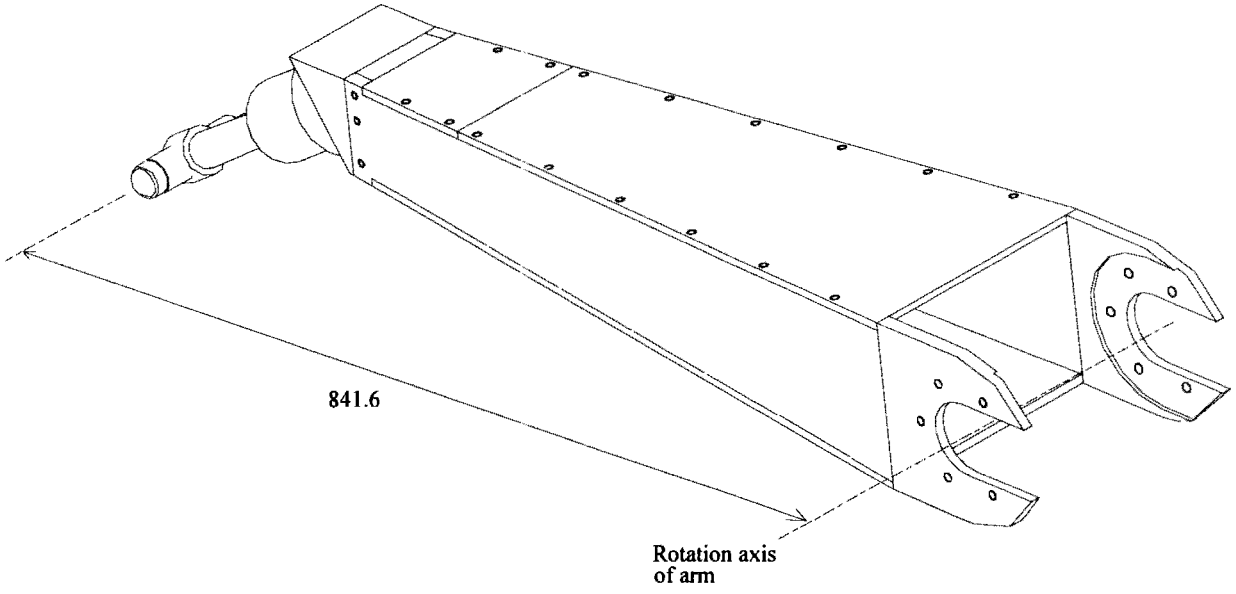


Figure 3.14 Upper arm with revolute joint attached.

The worst loading scenario experienced by the arms was bending about an axis perpendicular to the rotational axis of the arm as shown in figure 3.15.

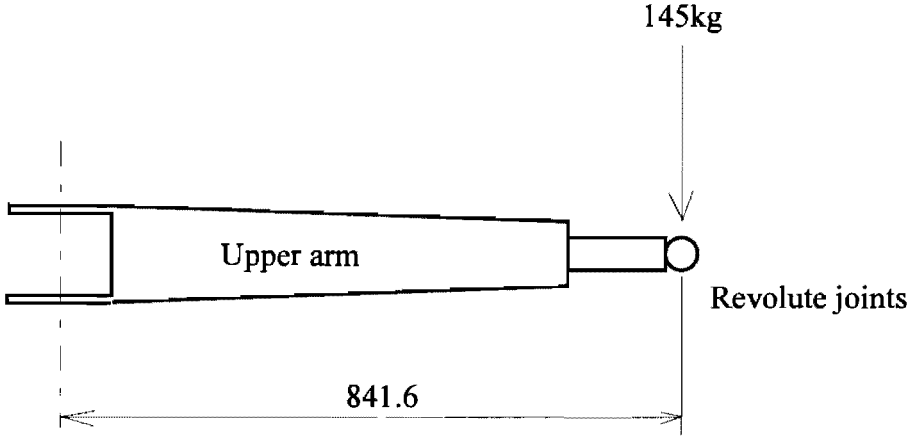


Figure 3.15 Maximum arm loading condition.

The loading model for the upper arms is shown in figure 3.16 below.

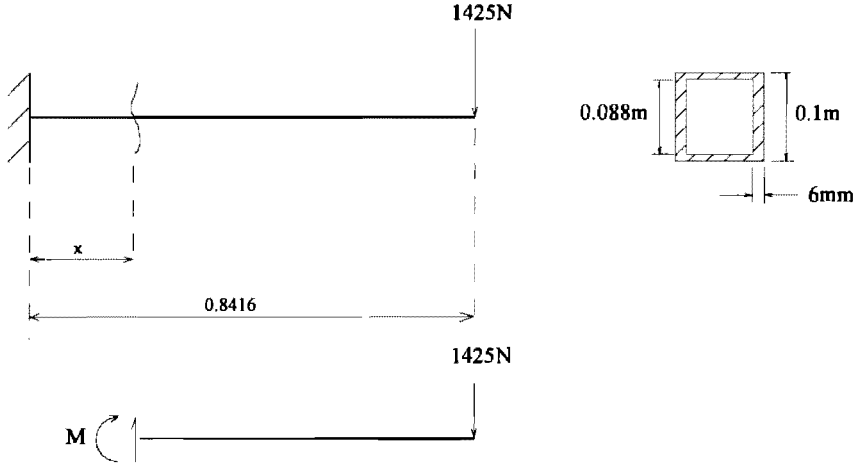


Figure 3.16 Loading model for the upper arms.

Summing the moments about M we get

$$\sum \mathrm{M} = 0: \quad \mathrm{M} + 1425(0.8416 - x) = 0$$
$$\mathrm{M} = -1425(0.8416 - x).$$

The largest moment M = -1200Nm occurs when x = 0. Assuming that the arm is uniform along its length, i.e. does not taper, the deflection of the arm due to the 145kg is then

$$w = \frac{Pl^2x}{2EI} - \frac{Pl^3}{6EI} \qquad I = \frac{a^4 - b^4}{12}$$
$$= \frac{1425 \times 0.8416^2 \times 0.8416}{2 \times 70 \times 10^9 \times 3.34 \times 10^{-6}} - \frac{1425 \times 0.8416^3}{6 \times 70 \times 10^9 \times 3.34 \times 10^{-6}}$$

$$= 1.2mm.$$

To decrease the angle achievable between the upper and lower arms, the revolute joint attached to the upper arm is offset as shown in figure 3.17 below.

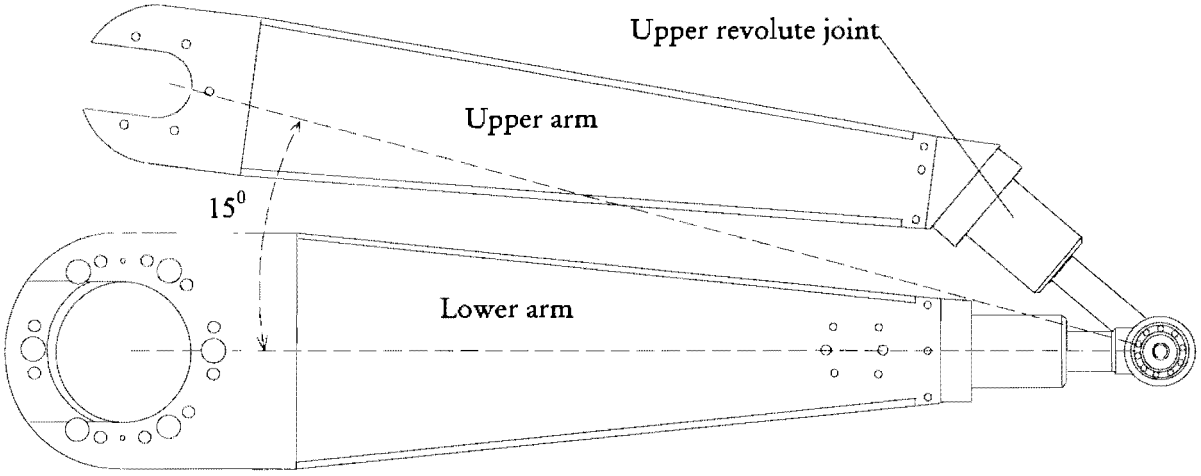


Figure 3.17 Reduced angular closure of the arms due to the offset of the upper revolute joint.

3.7 Design of the revolute joints

Two options are available for the construction of the spherical joints at the end of the arms: either a ball joint or three intersecting revolute joints can be used. The ball joint option was ruled out since the ball joints have a tendency to bind under tensile loads. The second option was to use 3 revolute joints connected serially to form a complex joint providing 3 rotational degrees of freedom as shown in figure 3.19.

A 3-2-3 rotational set was chosen since this configuration provides a fairly simple design. The singularity occurs when the axes of the revolute joints are aligned as shown in figure 3.18 and so the complex joint loses a degree of freedom. This position is the limiting position or angle to which the axes of the revolute joints may approach. This singularity, although in the outer limits of the work space of the robot, was not of great concern since it was well away from the necessary work space for beam-aiming applications.

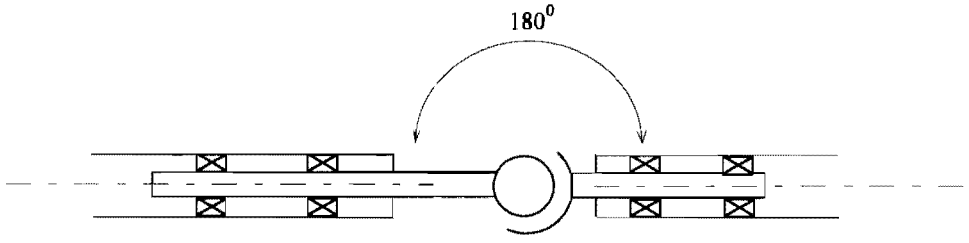


Figure 3.18 Revolute joints in a singularity.

A disadvantage with the revolute joint option as opposed to the ball joint option is that not only must the joints be assembled in the correct positions but their axes must also be angularly aligned, whereas the ball joint needs only to be positioned correctly. The combined weight of the revolute joints is 6.47 kg.

3.7.1 Design of the revolute joints in more detail

The revolute joints were designed so that they could be easily and accurately attached to the arms. The attachment flanges of the joints were designed to be bolted to the arms. They included two location holes for dowel pins and an optional hole for a spigot type location. Preloaded taper roller bearings were used so that there would be no clearance in the joints.

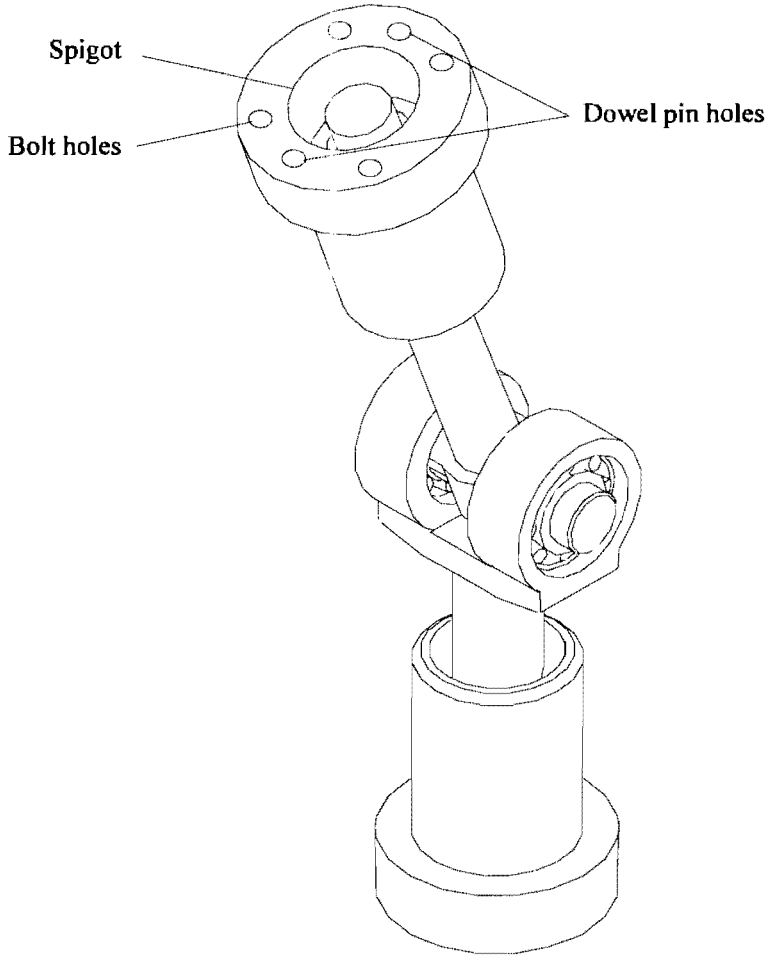


Figure 3.19 Three revolute joints in series forming a three dof complex joint.

The revolute joint attached to the upper arm is the most highly stressed of the three joints since it has the longest yoke, with the major stress due to bending.

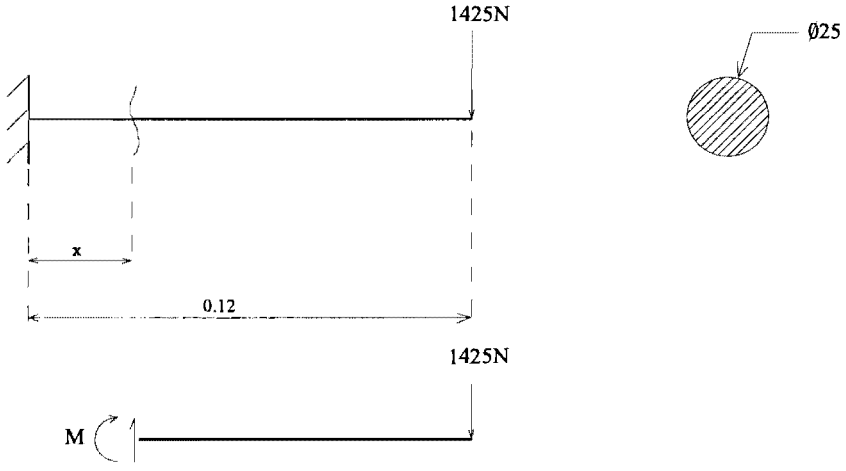


Figure 3.20 Loading model for the upper revolute joint.

The moments can be summed about M:

$$\sum \mathrm{M} = 0: \quad \mathrm{M} + 1425(0.12 - x) = 0$$
$$\mathrm{M} = -1425(0.12 - x).$$

The largest moment M = -171Nm occurs when x = 0. The maximum bending stress is then

$$\sigma_b = \frac{My}{I} = \frac{-171 \times 0.0125}{3.07 \times 10^{-7}}$$
$$= 7\,Mpa.$$

This stress is well below the yield strength (200MPa) of mild steel. The deflection of the yoke due to the 145kg load is given by

$$w = \frac{1425 \times 0.12^2 \times 0.12}{2EI} - \frac{1425 \times 0.12^3}{6EI}$$
$$= 0.013mm.$$

3.8 Design of the platform

The platform was designed to provide a very rigid mounting structure to which the upper arms could be accurately attached thus ensuring that the arms would rotate about well-aligned axes. It also provides an accurate surface for mounting a ball joint should the central strut be used.

The platform shown in figure 3.21 was designed to locate accurately the three tubes used to hold the upper arms. A 12mm sheet of aluminium was machined on an indexing table so that the highest accuracies could be achieved. The mounting blocks used to contain the shafts were located using dowel pins. The shafts were in turn also located using dowel pins thus providing precise axial positioning. The tubes were constructed from aluminium tubing and were attached to the shafts via pre loaded taper roller bearings. The passive arms could then be easily and accurately bolted to the tubes. The weight of the platform is 19.57 kg.

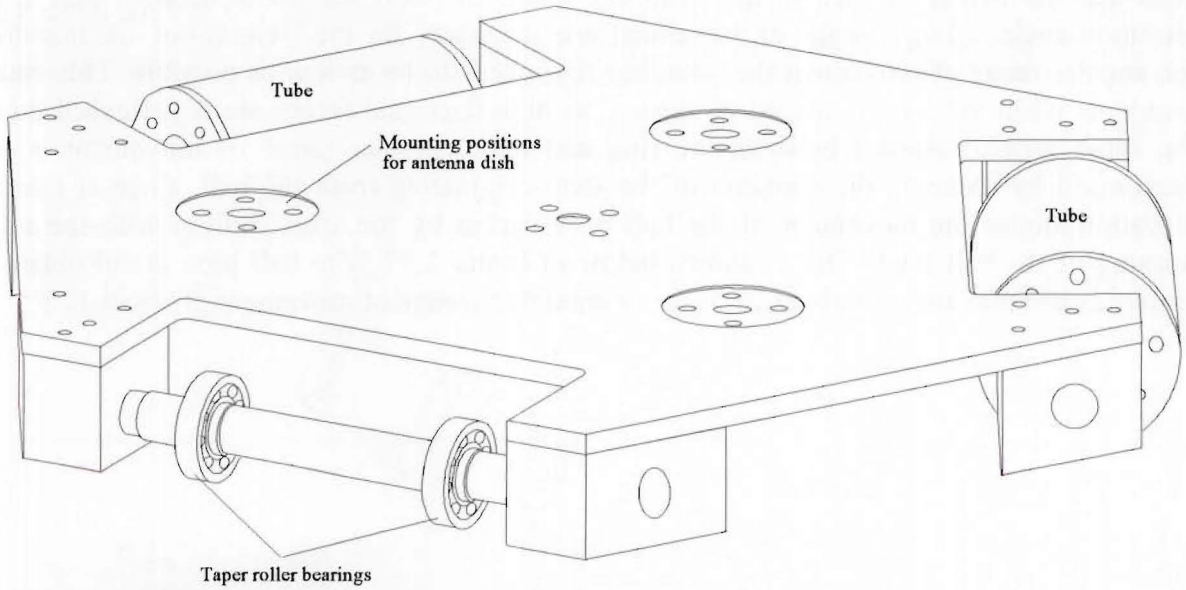


Figure 3.21 Platform showing attachment tubes for upper arms.

3.9 Design of the ball joints

Ball joints are used to attach the central strut to the base and platform. They have a steel housing with brass wear inserts. An outer housing is screwed onto an inner housing so that the clearance between the ball and inserts can be adjusted. To ensure concentricity a neat sliding fit was used between the inner and outer housings. The joint has accurate mating surfaces for attachment to external components.

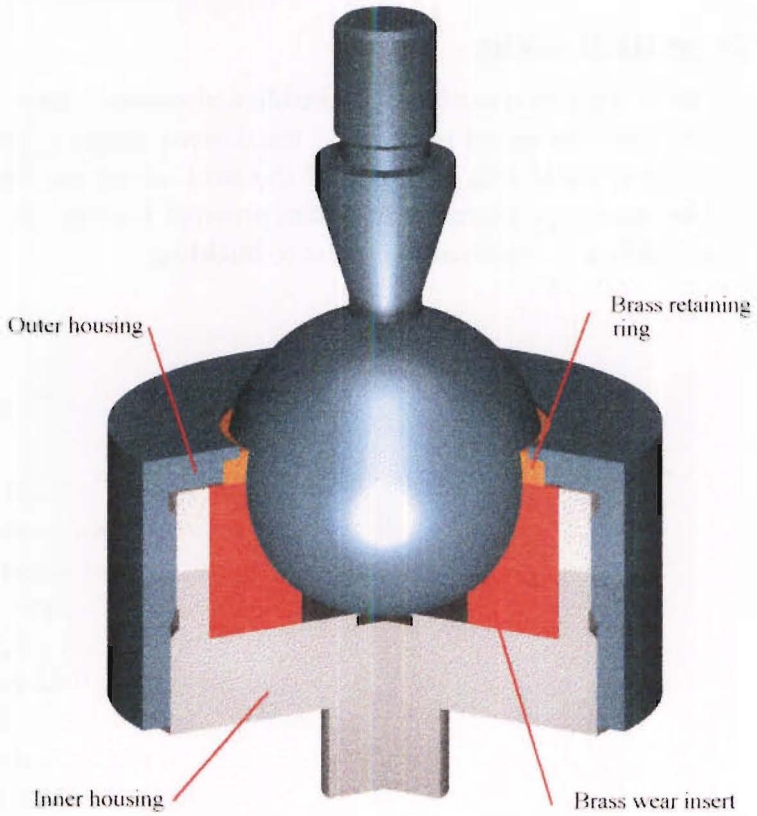


Figure 3.22 Part cut away of ball joint.

Although the ball joint used in the three dof spherical robot needed to achieve only a 45° elevation angle, a larger range of movement was necessary for the Delta robot. To maximise the angular range of movement the retaining ring needs to be as low as possible. This causes problems when the joint is placed in tension, as high frictional forces are experienced due to the small area of contact between the ring and ball. Angular range of movement is also maximised by reducing the diameter of the stem originating from the ball, since at extreme elevation angles the movement of the ball is restricted by the stem fouling with the upper housing of the ball joint. This is shown below in figure 3.23. The ball joint could obtain an extreme elevation angle of about 27.5°, i.e. a maximum range of movement of about 125°.

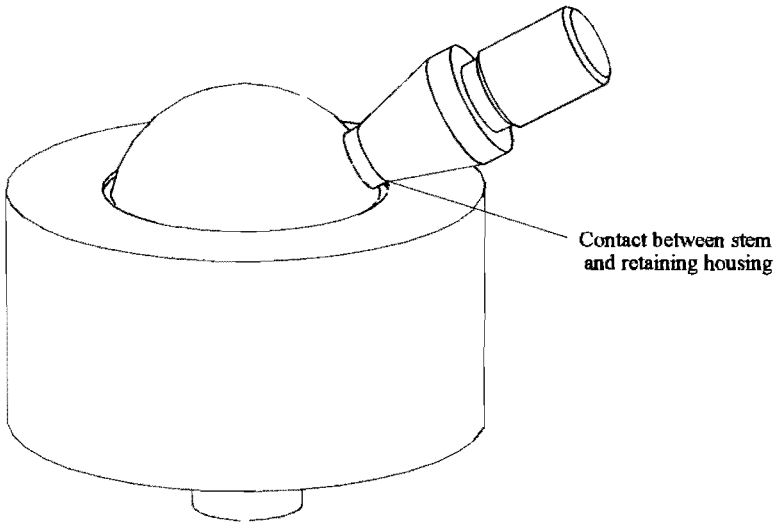


Figure 3.23 Ball joint with stem at extreme elevation angle.

3.10 Design of the central strut

The central strut was made from a nylon rod that slides inside a aluminium tube. An adjustable ring lock compresses the tube onto the nylon rod so that the desired length of the strut can be set. Two ball joints can then be screwed into each end of the strut which can then be attached to the base and platform. The strut experiences mainly compressive loading and since it has a high length to diameter ratio, failure is expected to be due to buckling.

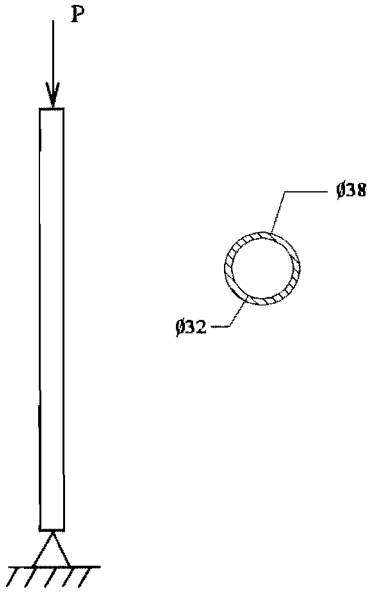


Figure 3.24 Loading model for central strut.

Assuming that the strut is made from aluminium tubing only, and assuming no eccentricity in the loading, the maximum buckling load sustainable is given by

$$P = \frac{\pi^2 EI}{l^2}$$
$$= \frac{\pi^2 \times 70 \times 10^9 \times 8.14 \times 10^{-7}}{0.9^2}$$
$$= 695kN.$$

This maximum possible loading is much higher than the expected loading.

3.11 Design of the end of travel limit switches

The safety of the robot, if anything untoward should occur, is very important since the harmonic drives are very costly to replace. There are number of possible control failure scenarios that need to be considered.

The worst situation that could arise is a failure of communication between the computer and the IFX controller during movement, or the IFX controller failing to act on commands in its buffer. Failure of communication could occur for a number of different reasons, such as a power failure to the computer, the computer 'crashing', or a connection coming loose. If a continuous command movement MC was being acted upon at the time of failure, the driven arms of the robot would continue driving until they collided with either the base or the upper arms. The sudden decelerations would cause very large forces to be exerted on the harmonic drives and more than likely cause severe damage to them.

Electromagnetic brake

The motors are available with an electromagnetic brake but at a much higher cost and so this option was excluded.

Sequence operation commands

If the computer is still communicating correctly, it is possible to use special sequence operation commands that can be activated by raising the sequence select pins on the IFX controller. The sequence could contain a set of commands to quickly decelerate the arms to rest. This sequence can be activated by a signal from a limit switch as when the driven arms travel past a predefined 'no go' area. Unfortunately because the controller must be told to continuously read the sequence select pins, the computer must still be able to communicate with the IFX controller, as it will read them only once upon powering up and will not read them again until it is commanded to do so. Therefore, even if a sequence select pin is raised high, it will not start a sequence unless the controller has been commanded to read the pins by the computer. There are added problems with this method since the command (Z) to read the sequence select pins clears the IFX buffer immediately and then reads the sequence pins but in doing so, destroys any following commands that may have been buffered. Since the Z commands would have to be continually sent to make the IFX check on the sequence pins, the movement commands in the IFX buffer will be continually deleted.

Another alternative (provided the computer is still functioning correctly) is to have the computer receive an 'end of travel limit' signal and then to send either a deceleration sequence of movement commands to the IFX, i.e. send V0 commands, or to command the IFX to run a predefined deceleration sequence using the X commands. Both options are unattractive since the IFX must first work its way through the other preceding commands in the buffer, and therefore catastrophe could have occurred before any deceleration had begun. Another possibility is to send the stop command S; although this command is an immediate command, (i.e. it will go straight to the head of the buffer queue and be acted upon immediately) the motors will only decelerate at the rate of their last acceleration commands. If the acceleration commands were low, the motor will take a long time to come to a stop. Ideally the IFX driver would have been able to act upon the sequence select pins as soon they were raised without having to be instructed by the computer to read them first.

End of travel limit switches

The final alternative is to design a fail safe method that uses the end of travel limit switch pins on the IFX controller. A reed switch (c.f. figure 3.25) was mounted to the base and a magnet mounted to the revolving torque tube. When the reed switch is activated by the magnet the limit switch pin is raised high bringing the motor to an immediate halt. This method has the advantage that it does not rely on the computer to function correctly. When these limit pins are shorted by the limit switch, the IFX energises only one coil and the motors are immediately decelerated to a halt. Unfortunately with higher inertial loads the motors will lose 'sync' and so the deceleration is only slightly quicker than if the power had been cut off completely to the motors.

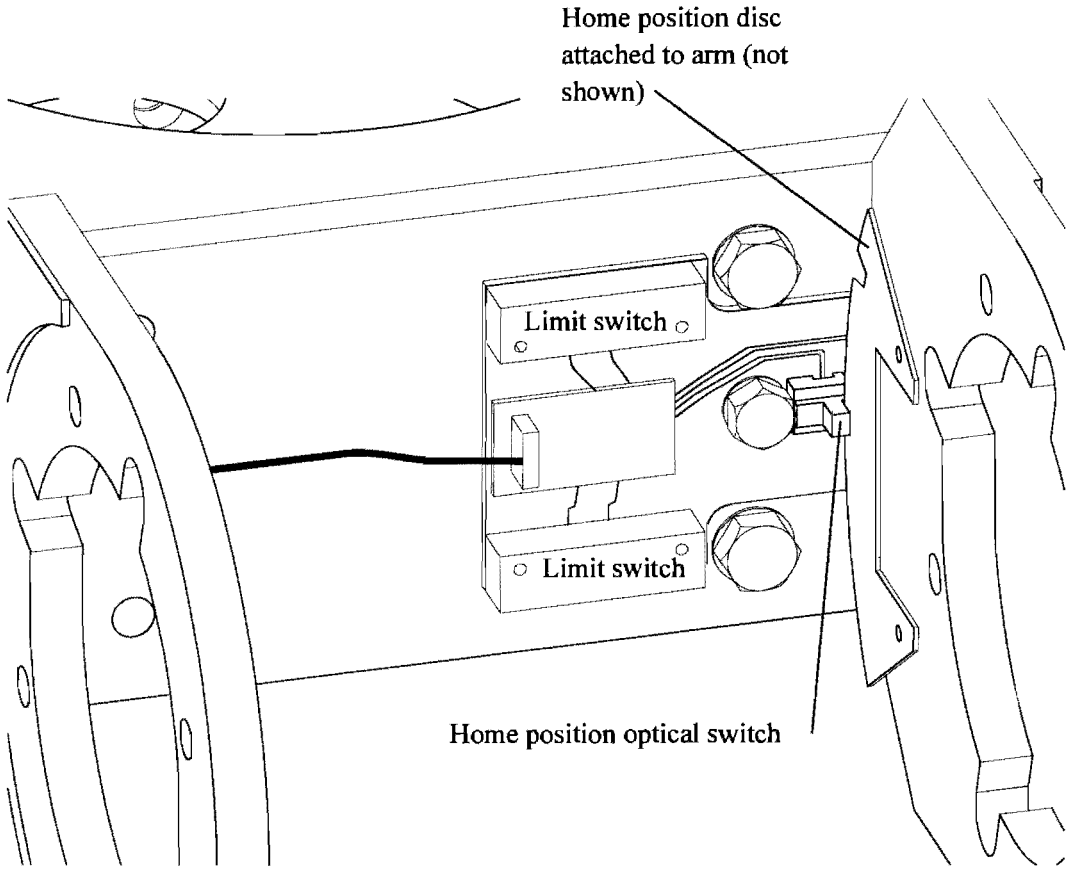


Figure 3.25 Limit switches and home positioning optical switch.

3.12 Design of the home position switches

Optical switches were used for a home positioning system for the arms. A disc mounted to the arms was used to cut the light beam. As soon as the selected edge is detected the home limit input on the motor controller is pulled low. The IFX will then decelerate to rest at the last previously defined acceleration. The IFX will then position the motor 1/32 of a rev on the outside of the selected edge. Finally, the motor will creep at 0.1 rps in the direction of the home active region; it will then stop, and the homing process is complete.

3.13 Design of the buffers

Rubber buffers were attached to the base to cushion the driven arms should they not come to a complete rest before hitting the base. The buffers ensured that the maximum design forces experienced by the main bearings and the splines in the gearboxes are not exceeded.

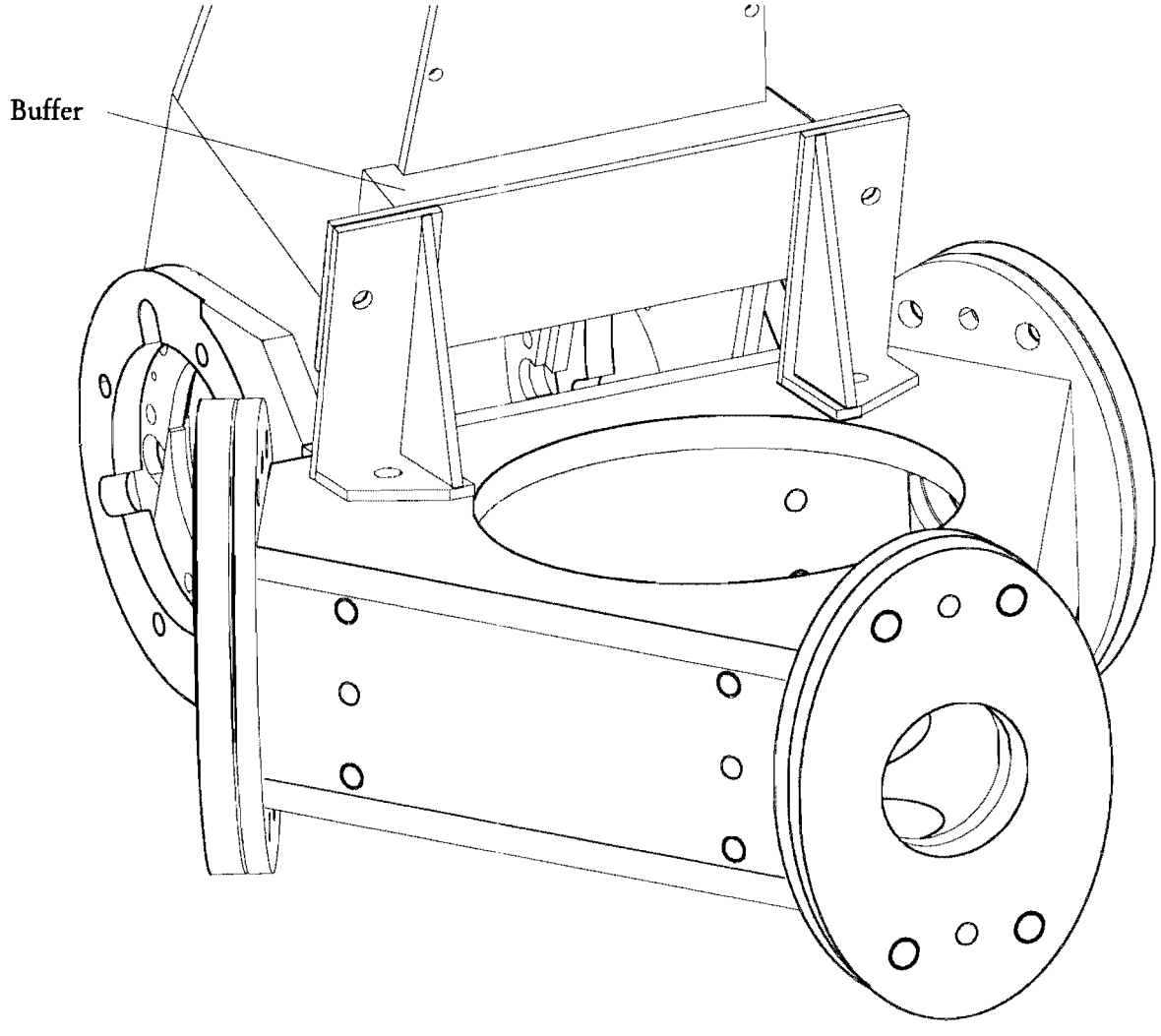


Figure 3.26 Buffer attached to the base to cushion the impact of the driven arm.

Chapter 4

Control of the robot

4.1 Introduction

This chapter describes the design and use of the robot control system. The bottom up modular design enables the system to be adapted to other robots, other motor controllers and different high level user interfaces. The program is interrupt driven in both transmission and reception so that the processor is freed to compute the desired robot movements even as communications between the computer and motor controller are on-going.

4.2 Description of the hardware

The three step motors used to drive the robot arms are controlled and driven by Digiplan IFX step motor drivers. An IBM PC is linked to the Digiplan controller which is sent ASCII commands via an RS-232 serial communication port. The command language is referred to by the manufacturer as 'X Code'. The Digiplan controller has no input/output data flow control, i.e. no 'handshaking' or 'XON/OFF' capabilities, and therefore the flow control must be handled by the computer software only. The computer relies entirely on interrupt signals from the serial communication port to 'announce' the reception or transmission of data, i.e. polled communication is not used. Figure 4.1 below shows the basic set-up of the system hardware.

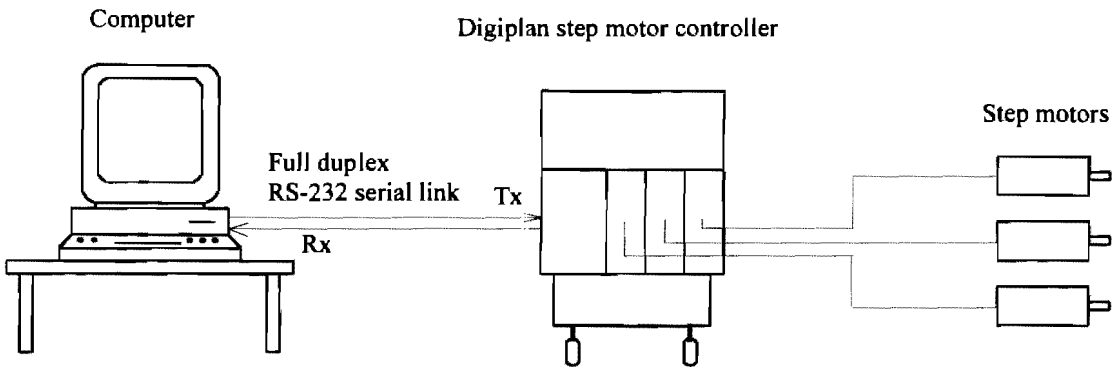


Figure 4.1 Physical set up of system hardware.

The serial link is full duplex, i.e. different lines are used for the two signal directions; hence with the use of this system, data can be transmitted and received at the same time. When a character is sent to the controller, the character is echoed back to the computer. The advantage of this scheme is that the user gets verification that the correct character was received by the controller.

Each Digiplan controller has a small buffer which can hold a series of X code commands not totalling more than 512 bytes. Obviously it is important that during any movements, these buffers neither empty or overflow. To ensure that this condition never occurs, an X code command '1BS' is regularly sent to the controller by the computer. The Digiplan controller then returns to the computer a value corresponding to the amount of space left in its buffer. This

value can then be used by the computer to maintain the Digiplan buffer at an acceptable level. To simplify this process only one buffer is interrogated as the X Code is generated according to a set of established rules and so each buffer should contain roughly the same amount of data.

Both the transmission and reception of data by the computer is interrupt driven to allow the computer to carry out other supporting tasks as well as the primary task of transmitting and receiving data. Some of these tasks include computation of the ways points when computing the path of the platform, generation of X Code for subsequent moves, and maintenance of buffers inside the computer memory.

4.2.1 The Parker Hannifin IFX motor controller

The Parker Hannifin Digi Plan IFX indexer is a multi-axis controller for step motors. The controller has a number of limitations:
1) No servo control, only endpoint verification (if an encoder is installed).
2) The control code is ill equipped for simultaneous coordination of multiple axes.
3) The serial communication link to the controller is primitive and awkward:

- No handshaking of any kind is supported so frequent buffer status requests must be made.
- The IFX can effectively only accept 1 status request at any time (this is discussed in more detail below).

The IFX controller has a 512 byte buffer to store incoming control code. An X code command sequence is a string of ASCII characters. The command set is divided into a number of classes of command:

Parameter - Commands that set parameters.
Execute - Commands that cause operations to be performed.
Sequence - Commands pertaining to stored sequences.
Status - Commands that cause the IFX to send information to the terminal.

Each of the above types is divided into two further categories:

Immediate - The IFX responds immediately.
Buffered - The IFX stores the command in its input buffer and executes it in turn.

A problem arises when handling status commands. When the IFX responds to a status command it will interleave its reply with other echoed characters, therefore rendering the reply useless. To ensure that this does not happen transmission from the PC must pause until the reply is received. The consequences of this are that if an immediate status command is sent such as a request for the current buffer status size (1BS), transmission must be halted until the reply is received. If a buffered status command is sent such as a request for the motor position (1PR), no further commands can be sent until the request is acted upon. Unfortunately in this case the buffer will then empty before further commands can be sent.

4.2.2 Coordinating the axes

In order to force simultaneous coordination of three axes, trigger inputs are used. This ensures that as each way point is reached, each motor has completed its movement before the move to the next way point is initiated. Each IFX has three trigger inputs and two outputs. The outputs are controllable with the X Code 'O' command. For example, the command 'O11' sets both outputs to logic 1. The trigger inputs are examined with the 'TR' command. The command 'TR0X0' causes the IFX to wait until the 1 and 3 trigger inputs are logic 0 regardless of the

state of the 2 input. Note that the trigger inputs are active low so that a signal from the outputs appears inverted at the trigger inputs. In the set up used here the outputs of each IFX controller are connected to the trigger inputs of the other two controllers, thus allowing each IFX to signal the others.

4.3 Control modes

To track a satellite accurately, the path of the platform needs to be controlled, i.e. the position and time taken to reach that position needs to be controlled. It is not possible to implement reliable velocity profiling. This is due to the large delay time between the PC requesting the current motor position and the motor acting on the new set of movement commands. But something approximating path control can be achieved by breaking the path down into discrete way points, and making the platform pass through these way points at the correct time.

The robot control modes can be divided into two separate categories, either point to point positioning, or a form of path control that 'integrates' many points (called way points) to form a path.

4.3.1 Point to point positioning

The point to point positioning control can be achieved in two ways. The first method drives all motors at their maximum accelerations and velocities, and therefore each motor (arm) will arrive at its desired end point position in the shortest possible time, but generally the motors (arms) will not arrive at their desired positions at the same time.

The second method introduces trapezoidal velocity profiling, i.e. the motors are ramped up and down at predetermined rates. Using this method, each arm will arrive at its end point position at the same time so that there will be less deviation from the ideal path as the platform moves from the start point to the desired end point. Each motor is given a particular acceleration and maximum 'ceiling' velocity so that although each arm will travel different distances, they will take the same amount of time to travel their respective distances. An example of velocity/time graphs for each arm is shown below in figure 4.2.

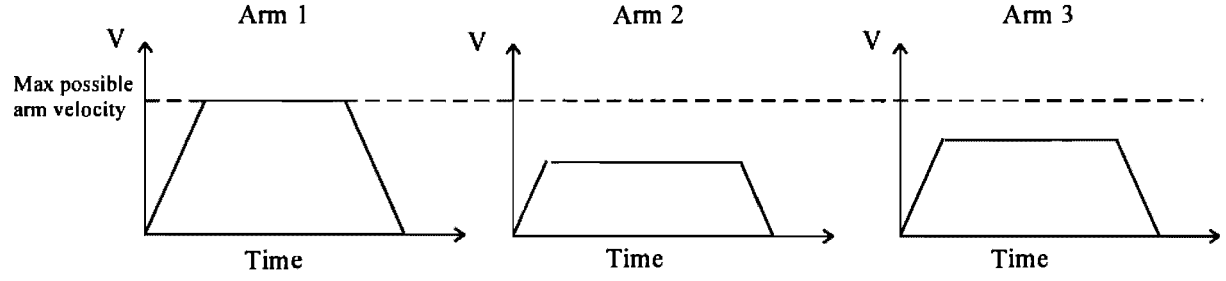


Figure 4.2 Trapezoidal velocity profiling for minimum time positioning.

The maximum ceiling velocity and acceleration is set by the characteristics of the motors driving the arms. The computer is able to calculate the required velocity for each arm by using the fact that the area under each velocity/time graph is equal to the distance each arm has to travel. The accelerations (the slopes of the lines) is preset to a desired value. If as in the case of arm 1, the ceiling velocity of the motor has been reached, the time for the movement must be increased. Consequently a warning message will appear on the screen notifying the user that the desired movement could not be achieved at the speed specified. The other arm velocity profiles are then scaled from the limiting trapezoid.

4.3.2 Path control

Path control is achieved by breaking down (or discretising) a complete path into many start and end points termed way points. Either method of point to point positioning can be used, with trapezoidal profiling producing less deviation from the ideal between each way point, hence this method will produce the most accurate complete path. The accuracy of the complete path is increased when the way points are spaced more closely.

4.4 Control program

The software was written in the C language. It was written from the bottom up, i.e. the routines are divided into a structured hierarchy of levels, each level calling a lower level through a clearly defined protocol. The design facilitates interchangeability of parts so that if, for example, a new motor controller is used then the driver section can be replaced leaving the rest of the program unchanged. Alternatively, a better high level programmer interface could be used and the driver level could remain unchanged as long as the programmer level interfaces correctly to the driver. The overall scheme is shown in figure 4.3.

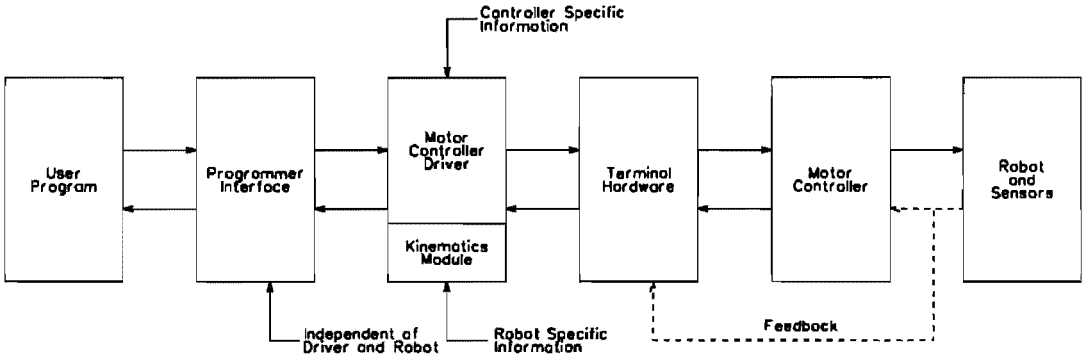


Figure 4.3 Block diagram showing overall layout of robot control system.

4.4.1 The structure of the program

The Parker Hannifin IFX driver program (ph_ifx.drv) features interrupt driven two way serial communications and echoes the overall bottom up program structure. A schematic of the program is shown in block diagram form in figure 4.4. A complete listing of the routines is included in Appendix F.

System level routines write directly to the Universal Asynchronous Receiver/Transmitter (UART) and Programmable Interrupt Controller (PIC). The DOS and BIOS serial communication services are bypassed.

Level 1 routines communicate indirectly with the system level through the buffers and status variables. The routines either insert or retrieve single characters from the buffers, interrogate the buffer structures, or allocate memory for the buffers. Other level 1 routines check the type of characters that are to be buffered.

Level 2 routines operate by recursively calling the level 1 routines. The routines buffer strings of characters and also write to and read from the status structure.

Level 3 routines are the routines that form the interface between the programmer level routines and the driver routines.

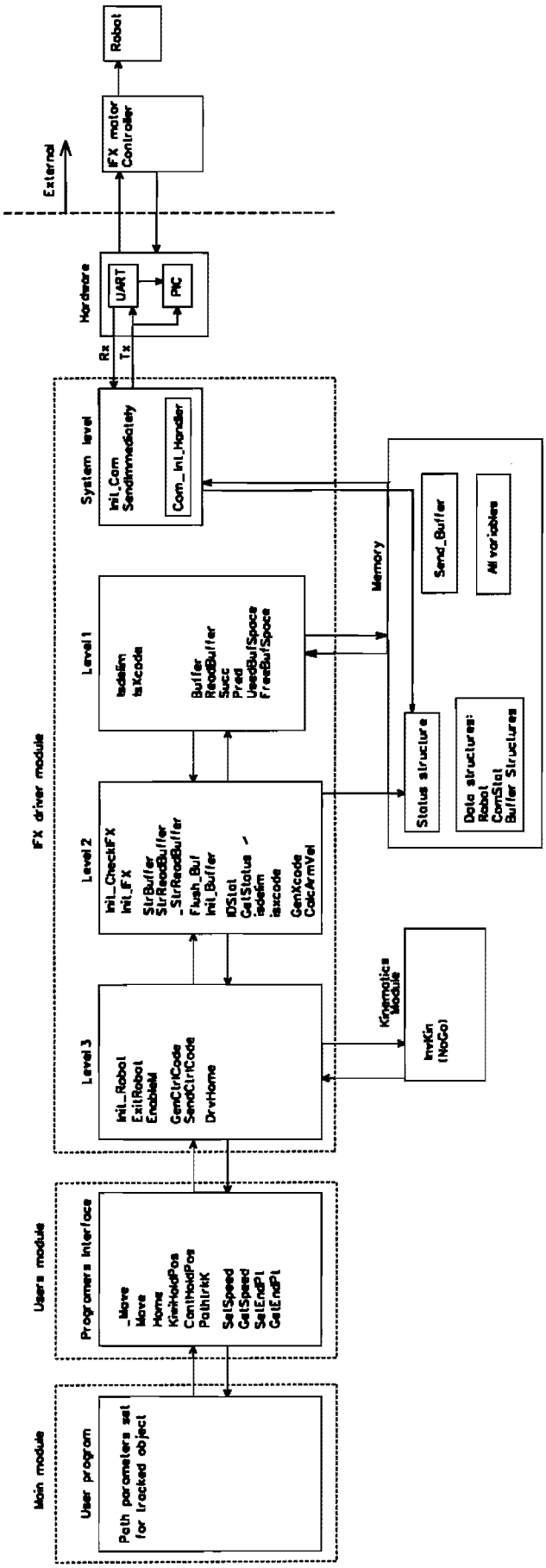


Figure 4.4 Detailed block diagram showing hierarchy of control system software.

The programmers interface forms a link between the programmer level routines and the driver routines. This module (user.c) calculates the movement commands necessary for a desired path to be followed.

The main module (demkiwi1.c and demcant1.c) allows the user to set the desired path parameters (c.f. section 7.1) such as the speed Ω, height (h) of orbit, the orbit offset angles (α_2 and α_3).

4.4.2 Data structures

The principal data structures are the buffers and the status structure. Various ancillary structures and type definitions exist to simplify the interface between code modules.

Buffer Structure - A buffer can have no more than 64K (the limit that the Turbo C compiler can allocate using malloc) and each buffer has a control structure defined with pointers to the base and top of the buffer, pointers to the entry and extraction positions of the data, and finally pointers to the size and status of the buffer. The buffer can be in one of three states; normal, empty or full. The buffer operates in a circular fashion with the pointers reset to the base once they have reached to top.

The function Init_Buffer allocates memory for the buffer and initialises the control structure. The driver has a global buffer named SEND_BUFFER which acts as a repository for movement data (X code) waiting for transmission. This buffer is automatically created when the function Init_Robot is called.

Status Structure - This structure holds all status information fed back from the motor controller. The structure exists in memory as an array of buffer control structures, each buffer control structure pointing to a small buffer also created when the function Init_Robot is called. The interrupt handler automatically updates the structure whenever a status command is issued.

ComStat - This holds variables pertaining to the serial communication flow. This structure is defined to show information on four different states which are:

Send	transmit-if-able status,
Emerg	emergency type (if any),
StatID	storage of status identifier (Com_Int_Handler), and
MovCnt	number of complete code sequences in SEND_BUFFER.

If the ComStat.Send variable is set to 'ON', this indicates that the interrupt handler will transmit characters from the SEND_BUFFER unless
a) the IFX buffer is full (XOFF state) or,
b) a status command has been sent and a reply has not yet been received.

If the ComStat.Send variable is set to 'OFF' a 'BC_PAUSE' has been encountered in the SEND_BUFFER and transmission is ceased.

4.4.3 The interrupt handler

Communications between the interrupt handler and the buffers

Since the communications interrupt handler is an interrupt routine (c.f. Appendix F.8 for a complete listing), arguments cannot be passed to it directly. The handler requires information about what type of data it is transmitting, and when to cease transmission. This information is passed to the handler through the SEND_BUFFER in the form of buffer control codes (BC-codes) which precede the buffer status commands. Thus the interrupt handler is then able to recognise if the code is a status command and of what type. These codes are characters that are not X Code. The interrupt handler intercepts these control codes as they are extracted from the SEND_BUFFER and sends a neutral X code character (a space) in their place. The three types of buffer control codes are

BC_PAUSE pause transmission (unconditional).

BC_IMMED immediate status command. These are commands which upon reception by the motor controller will be acted on immediately regardless of the code residing in its buffer.

BC_BUFFR buffered status command. These are commands which upon reception by the motor controller will be placed at the end of its buffer and acted on only when it has processed the preceding code residing in its buffer.

Information received by the interrupt handler from the motor controller such as its buffer size status is stored in memory by the interrupt handler into the appropriate buffer which can then be accessed by higher level routines.

Detailed operation of the interrupt handler routine

The routine responds to UART interrupts (specifically - transmit (Tx) and receive (Rx) interrupts), by reading the Interrupt Identification Register (IIR) and switching to an appropriate part of the function depending on the interrupt type. Data reception is only enabled when a status command has been issued so that the reply can be captured. Normally echoed characters are ignored.

An interrupt must be cleared before the handler can be exited, and because multiple interrupts can occur simultaneously (or in quick succession), the handler routine is enclosed in a loop that checks the 'Pending' bit of the IIR until it goes high (indicating that there are no more interrupts to be handled).

An interrupt is cleared by writing to, or reading from, the data register (DATAPORT). If a character is received then the port must be read, and if the transmit shift register is empty then the port must be written to. The other way to clear an interrupt is to disable and re-enable the appropriate bit in the Interrupt Enable Register (IER). If this is done inside the interrupt handler the program will become stuck in the interrupt handler and never exit. It must be re-enabled from outside the interrupt handler.

To avoid the problem of the IFX driver sending characters after a status command has been transmitted, the communications interrupt handler routine is written so that transmission is paused whenever a status command is sent. Each status command in the SEND_BUFFER must be preceded by an identifying code (a buffer control code or 'BC-code') so that the communications interrupt handler routine can identify it.

Once the interrupt handler has intercepted a BC-code the next few characters comprising the status command are sent until a space character is encountered indicating the end of the status

command reply; then transmission is disabled. When intercepted the BC-code is converted into an index in the status array so that the reply may be stored in the appropriate buffer, e.g. the reply for a 1BS command would be stored in the buffer size status array. The communications interrupt handler then enables reception of the characters until a carriage return (indicating the end of the status command reply) is received, then reception is disabled and transmission is resumed.

Of great importance to the program is the frequently used buffer size status command '1BS'. This command returns the amount of free space inside the first IFX buffer. This information is used to implement a type of artificial XON/XOFF flow control. If the reported free space is less than a certain amount (200 characters) then transmission is paused and a series of '1BS' commands are sent until the IFX reports that the free space is above a certain amount (400 characters). Transmission is resumed once the free space is large enough.

4.5 Running the robot

To run a movement the following procedure should be carried out:

1) Ensure the following connections to and from motor controller are hooked up:
   a) the serial connection from the computer to the controller,
   b) the limit switch connection from the robot to the controller, and
   c) the power connections and earth from the motors to the controller.

2) Power up the motor controller.

3) Make a project of the following files:
   Demkiwi1.c (or Demcant1.c), Kiwi.kin (or Cantrk.kin), User.c and Ph_ifx_drv.

4) Define the robot type in User.c.

3) Set the desired parameters for movement in Demokiwi1.c or Democant1.c.

4) Compile and run the code, which will request the user to start the movement.

4.6 Some observations about the real moving robot

The movements of the robot are rather jerky since the robot must start and stop at each way point. As explained in section 4.3 this jerky movement is due to the fact that velocity profiling could not be implemented with this particular motor controller. Apart from the jerky movements the robot was able to demonstrate full coverage of the upper hemisphere for both the two and three dof configurations, although the two dof configuration showed significant compliance at low elevations angles. This problem is discussed further in section 10.3.

Chapter 5

Geometric analysis

5.1 Introduction

This chapter investigates some solutions to the position kinematics of the two and three dof mechanisms. For beam aiming applications the concern is not so much with the position of the platform, but rather with the orientation of the platform as that will dictate the pointing direction of the antenna. However for the three dof mechanism, the third redundant translational freedom may be useful for maximising the stiffness of the manipulator.

The forward kinematics of a robot determines the position and orientation of the end effector or platform given the position of the independent (or actuated) joints of the robot. For beam aiming applications in particular, the problem becomes, given the position of the actuated joints, what is the pointing direction of the platform.

The inverse kinematics of a robot determines the position of the independent or actuated joints given the position and orientation of the end effector or platform of the robot. For beam aiming applications in particular, the problem becomes, given the pointing direction of the platform, what is the position of the actuated joints.

For the three dof mechanism the translational freedom can be specified by a restricted range of the cartesian coordinates or by the plunging/radial distance between the platform and base. For most applications the plunging distance is probably the most natural freedom to choose.

In the following sections a closed form solution is obtained for the direct and inverse problems for both the two and three dof mechanisms.

5.2 Direct kinematics for the three dof mechanism

Section 5.2.1 is concerned with solving the direct kinematics for the three dof mechanism with generalised geometry. Sections 5.2.2 and 5.3.3 deal with particular cases of geometry of the mechanism that reduce the complexity of the kinematic equations and also reduce the number of solutions to these equations.

A note on the notation of points and vectors

To simplify the notation both the position vector of a point and the point itself will be written as the same character. If the point in question, e.g. B is being referred to as a vector quantity it shall be denoted in bold, e.g. $\boldsymbol{B}$. The same is true for the coordinate axes, (e.g. u), if the axes is defined by some direction vector it shall be denoted in bold, e.g. $\boldsymbol{u}$.

5.2.1 Direct kinematic solution for a mechanism with general geometry

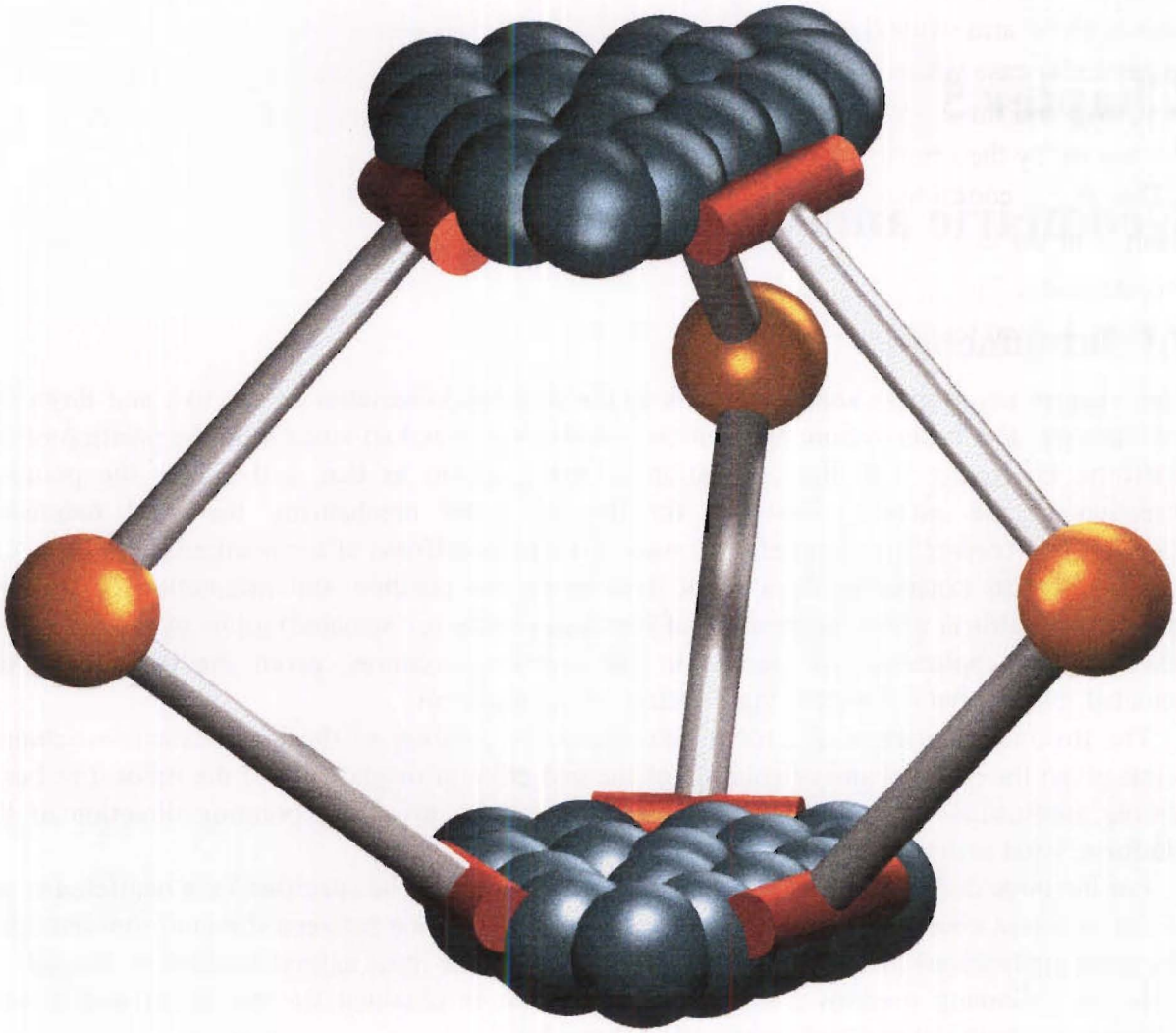


Figure 5.1 General three degree-of-freedom mechanism.

The forward kinematics of Stewart platforms have been extensively studied in recent years: Charentus and Renaud [1989], Nanua et al [1990], Parenti-Castelli and Innocenti [1990], Zhang and Song [1991], Merlet [1993], Lin *et al.* [1994], Zhang and Song [1994], Chen and Song [1994], Nair and Maddocks [1994], Wen and Liang [1994], Sreenivasan *et al.* [1994], Dasgupta and Mruthyunjaya [1994], Wampler [1994]. In some earlier publications Nanua *et al.* [1990], Parenti-Castelli and Innocenti [1990] the kinematic equations are derived by modelling the Stewart Platform as a triple arm mechanism i.e. each pair of spherical joints along one side of the base or platform is represented as a single revolute joint. The resulting triple arm mechanism is in fact one portion (upper or lower) of the three dof mechanism shown in figure 5.1.

Kinematic model of lower portion of mechanism

The principal coordinate system is termed the $\mathcal{F}_0$ system (c.f. figure 5.2). For the particular case where the points P_1, P_2 and P_3 form an equilateral triangle and the y_n axes (revolute axes) are coplanar with the equilateral triangle, the origin O_b is chosen to be at the centroid of the points P_1, P_2 and P_3.

The $\mathcal{F}_n$ (n=1,2,3) coordinate systems are attached with their origins at the centre of the driven revolute joints P_n which are at positions $(a_n, b_n, c_n)^T$. The x_n and z_n axes lie in the plane of rotation of the arm while the y_n axes lie along the rotation axes of the driven revolute joints. For the particular case where the y_n axes are coplanar, the z_n axes are normal to the plane formed by the y_n axes and the x_n axes are also coplanar with the y_n axes. The angle β_n is the variable angle which is set by the actuator for each arm.

The $\mathcal{F}_{u_1 v_1 w_1}$ coordinate system has its origin attached to the end of arm 1 at the point B_1 which is at the centre of the spherical joint (or the intersection point of the equivalent three revolute axes). The $\boldsymbol{u}_1$ axis is defined by the direction vector $\overline{B_3B_1}$. The $\boldsymbol{w}_1$ axis is normal to the plane defined by the points B_1, B_2 and B_3, and $\boldsymbol{v}_1 = \boldsymbol{w}_1 \times \boldsymbol{u}_1$.

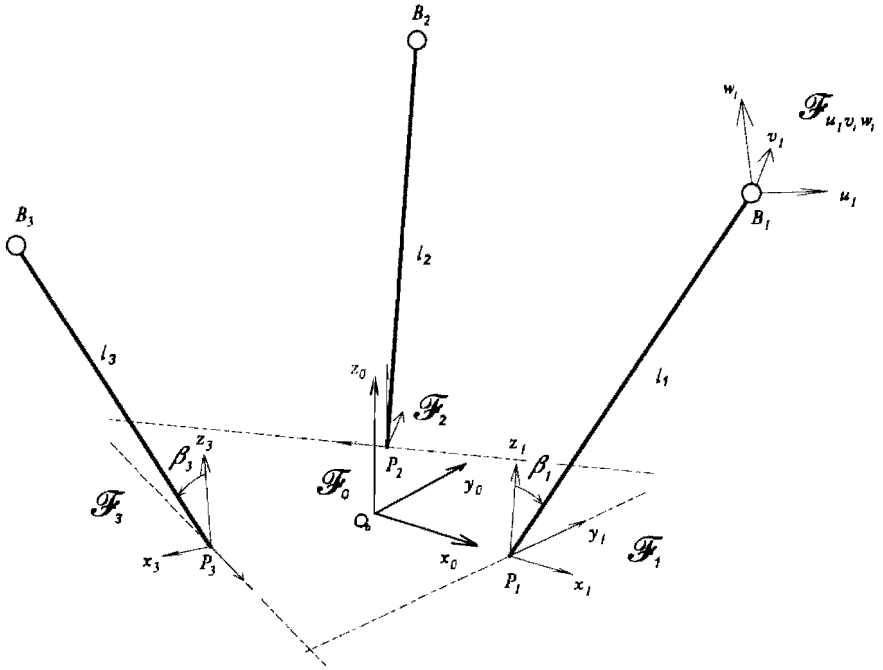


Figure 5.2 Lower portion of the mechanism.

Transformation matrices used for transforming from the $\mathcal{F}_n$ systems to the $\mathcal{F}_0$ frame

The 4×4 homogeneous transformation matrices ${}^0\mathbf{T}_n$, transforming from a point expressed in the $\mathcal{F}_n$ systems to the $\mathcal{F}_0$ frame, can be represented by a combination of three rotations and three displacements. The resulting matrix can be written as

$$ {}^0\mathbf{T}_n = \mathbf{T}_{z_n,c_n}\mathbf{T}_{y_n,b_n}\mathbf{T}_{x_n,a_n}\mathbf{R}_{z_n,\phi_n}\mathbf{R}_{y_n,\theta_n}\mathbf{R}_{x_n,\psi_n} $$

$$ = \begin{bmatrix} c\phi_n c\theta_n & c\phi_n s\theta_n s\psi_n - s\phi_n c\psi_n & c\phi_n s\theta_n c\psi_n + s\phi_n s\psi_n & a_n \\ s\phi_n c\theta_n & s\phi_n s\theta_n s\psi_n + c\phi_n c\psi_n & s\phi_n s\theta_n c\psi_n - c\phi_n s\psi_n & b_n \\ -s\theta_n & c\theta_n s\psi_n & c\theta_n c\psi_n & c_n \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where c = cos and s = sin.

The position of the points B_1, B_2 and B_3 expressed in the local systems $\mathcal{F}_n$ can be written as

$$ {}^{n}\boldsymbol{B}_n = \begin{pmatrix} l_n \sin\beta_n \\ 0 \\ l_n \cos\beta_n \\ 1 \end{pmatrix}. $$

By multiplying the points ${}^{1}\boldsymbol{B}_1$, ${}^{2}\boldsymbol{B}_2$ and ${}^{3}\boldsymbol{B}_3$ by the transformation matrices ${}^{0}\mathrm{T}_1$, ${}^{0}\mathrm{T}_2$ and ${}^{0}\mathrm{T}_3$ respectively, the points can be expressed in the $\mathcal{F}_0$ frame as

$$ \boldsymbol{B}_n = {}^{0}\mathbf{T}_n \, {}^{n}\boldsymbol{B}_n . $$

Note, points expressed in the principal $\mathcal{F}_0$ frame will not be preceded by a superscript, i.e. $\boldsymbol{B}_n = {}^{0}\boldsymbol{B}_n$.

Kinematic model of upper portion of mechanism

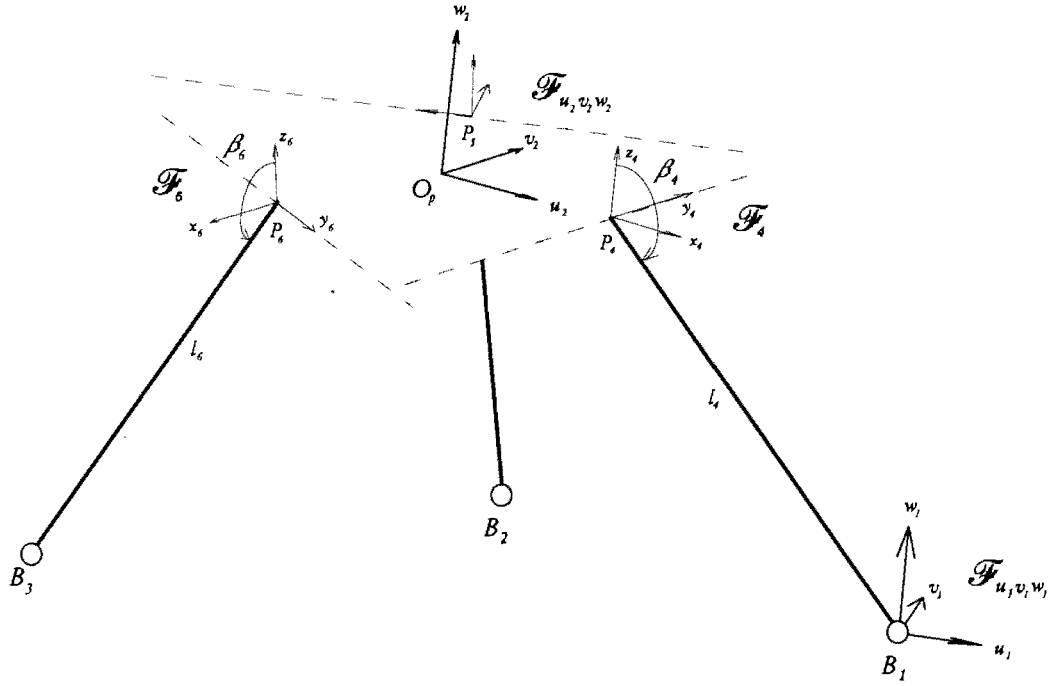


Figure 5.3 Upper portion of the mechanism.

The $\mathcal{F}_m$ (m=4,5,6) coordinate systems are attached with their origins at the centre of the passive revolute joints P_m. The x_m and z_m axes lie in the plane of rotation of the arm while the y_m axes lie along the rotation axes of the passive revolute joints. For the particular case where the y_m axes are coplanar, the z_m axes are normal to the plane formed by the y_m axes and the x_m axes are also coplanar with the y_m axes.

The $\mathcal{F}_{u_2v_2w_2}$ coordinate system is attached to the platform. For the particular case where the points P_4, P_5 and P_6 form an equilateral triangle and the y_m axes (revolute axes) are coplanar with the equilateral triangle, the origin O_p will be at the centroid of the points P_4, P_5 and P_6.

Transformation matrices used for transforming from the $\mathcal{F}_m$ systems to the $\mathcal{F}_{u_2v_2w_2}$ frame

The transformation matrices ${}^{u_2}\mathbf{T}_m$, transforming from a point expressed in the $\mathcal{F}_m$ systems to the $\mathcal{F}_{u_2v_2w_2}$ frame, can be represented by a combination of three rotations and three displacements. The resulting matrix can be written as

$$
{}^{u_2}\mathbf{T}_m = \mathbf{T}_{z_m,c_m}\mathbf{T}_{y_m,b_m}\mathbf{T}_{x_m,a_m}\mathbf{R}_{z_m,\phi_m}\mathbf{R}_{y_m,\theta_m}\mathbf{R}_{x_m,\psi_m}
$$

$$
= \begin{bmatrix} c\phi_m c\theta_m & c\phi_m s\theta_m s\psi_m - s\phi_m c\psi_m & c\phi_m s\theta_m c\psi_m + s\phi_m s\psi_m & a_m \\ s\phi_m c\theta_m & s\phi_m s\theta_m s\psi_m + c\phi_m c\psi_m & s\phi_m s\theta_m c\psi_m - c\phi_m s\psi_m & b_m \\ -s\theta_m & c\theta_m s\psi_m & c\theta_m c\psi_m & c_m \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_{m_x} & s_{m_x} & a_{m_x} & P_{m_x} \\ n_{m_y} & s_{m_y} & a_{m_y} & P_{m_y} \\ n_{m_z} & s_{m_z} & a_{m_z} & P_{m_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (5.1)
$$

The position of the points B_1, B_2 and B_3 expressed in the local systems $\mathcal{F}_m$ can be written as

$$
{}^{m}\boldsymbol{B}_n = \begin{pmatrix} l_m \sin\beta_m \\ 0 \\ l_m \cos\beta_m \\ 1 \end{pmatrix}.
$$

By multiplying the points ${}^{4}\boldsymbol{B}_1, {}^{5}\boldsymbol{B}_2$ and ${}^{6}\boldsymbol{B}_3$ by the transformation matrices ${}^{u_2}\mathbf{T}_4, {}^{u_2}\mathbf{T}_5$ and ${}^{u_2}\mathbf{T}_6$ respectively, the points can be expressed in the $\mathcal{F}_{u_2v_2w_2}$ frame as

$$
{}^{u_2}\boldsymbol{B}_n = {}^{u_2}\mathbf{T}_m \; {}^{m}\boldsymbol{B}_n . \quad (5.2)
$$

Solution of the forward kinematics

In order to solve for the relative displacement and orientation between the base and platform, the position and orientation of the $\mathcal{F}_{u_2v_2w_2}$ system must be determined with respect to the $\mathcal{F}_m$ system in terms of the three actuated dofs β_n. The closure equations are written for both the lower and upper portions of the mechanism. The three closure equations can be formed by noting that the distance between the points $\boldsymbol{B}_n$ is the same for both the upper and lower portions of the mechanism, i.e.

$$
\left|\boldsymbol{B}_{n+1} - \boldsymbol{B}_n\right| = \left|{}^{u_2}\boldsymbol{B}_{n+1} - {}^{u_2}\boldsymbol{B}_n\right| \quad (5.3)
$$

where n = 1,2,3 and if n+1=4 then n+1=1.

The subsequent squaring of the closure equations (5.3) leads to

$$
e^2, f^2, g^2 = \left|{}^{u_2}\boldsymbol{B}_{n+1} - {}^{u_2}\boldsymbol{B}_n\right|^2 \quad (5.4)
$$

where the distances e, f and g between the spherical joints are

$$
e = \left|\boldsymbol{B}_2 - \boldsymbol{B}_1\right| \quad f = \left|\boldsymbol{B}_3 - \boldsymbol{B}_2\right| \quad g = \left|\boldsymbol{B}_1 - \boldsymbol{B}_3\right|.
$$

After substituting equation (5.2) into (5.4) and noting that for the components of the matrices given in equation (5.1) the following identities apply for the unit normals

$$
\hat{\boldsymbol{n}}_m^T\hat{\boldsymbol{n}}_m = 1, \quad \hat{\boldsymbol{s}}_m^T\hat{\boldsymbol{s}}_m = 1, \quad \hat{\boldsymbol{a}}_m^T\hat{\boldsymbol{a}}_m = 1, \quad \hat{\boldsymbol{s}}_m^T\hat{\boldsymbol{a}}_m = 0, \quad \hat{\boldsymbol{s}}_m^T\hat{\boldsymbol{n}}_m = 0, \quad \hat{\boldsymbol{a}}_m^T\hat{\boldsymbol{n}}_m = 0.
$$

so that the following three closure equations are obtained

$$d_1 c\beta_4 + d_2 c\beta_5 + d_3 s\beta_4 + d_4 s\beta_5 + d_5 c\beta_4 c\beta_5 + d_6 c\beta_5 s\beta_4 + d_7 c\beta_4 s\beta_5 + d_8 s\beta_4 s\beta_5 + d_9 = 0, \quad (5.5)$$

$$h_1 c\beta_5 + h_2 c\beta_6 + h_3 s\beta_5 + h_4 s\beta_6 + h_5 c\beta_5 c\beta_6 + h_6 c\beta_6 s\beta_5 + h_7 c\beta_5 s\beta_6 + h_8 s\beta_5 s\beta_6 + h_9 = 0, \quad (5.6)$$

$$l_1 c\beta_6 + l_2 c\beta_4 + l_3 s\beta_6 + l_4 s\beta_4 + l_5 c\beta_6 c\beta_4 + l_6 c\beta_4 s\beta_6 + l_7 c\beta_6 s\beta_4 + l_8 s\beta_6 s\beta_4 + l_9 = 0, \quad (5.7)$$

where the constants d_i, h_i and l_i are derived in appendix B.

Numerical solution

A numerical solution method can be employed to iterate to the unknown angles given a suitable initial guess for the three angles β_4, β_5 and β_6. Since an analytical derivative is readily available, Newton-Raphson's method is computationally very efficient. This solution method has some disadvantages in that it does not lead directly to multiple solutions and is sensitive to the initial guess, especially for geometries close to a singularity.

Closed form solution

A closed form solution to the closure equations has been found which provides extra information about the mechanism. This closed form solution is able to provide all the configurations possible for the mechanism for each particular set of driven arm angles β_n. This information is particularly useful for the design of the real mechanism for beam aiming applications since the designer can ensure that no singularities are present in the operational work space. Control is then straight forward.

A direct solution to the closure equations (5.5), (5.6) and (5.7) can be obtained by expressing each of the equations in terms of a single variable. The decoupling of the closure equations involves the elimination of two of the arm angles β_i, resulting in a 16th degree polynomial expression in a single arm angle β. To facilitate this, substitute the following half tangent trigonometric identities into the closure equations.

$$c\beta_m = \frac{1-t_m^2}{1+t_m^2}, \quad s\beta_m = \frac{2t_m}{1+t_m^2}, \qquad \text{where} \quad t_m = tan\left(\frac{\beta_m}{2}\right).$$

With some algebraic manipulation the three closure equations can be written as three quadratics in terms of t_m. The first quadratic is

$$(g_2 t_5^2 + g_1 t_5 + g_0)\, t_4^2 + (f_2 t_5^2 + f_1 t_5 + f_0)\, t_4 + (e_2 t_5^2 + e_1 t_5 + e_0) = 0 \quad (5.8)$$

where

$$\begin{aligned} e_0 &= d_1 + d_2 + d_5 + d_9 & f_0 &= 2d_3 + 2d_6 & g_0 &= -d_1 + d_2 - d_5 + d_9 \\ e_1 &= 2d_4 + 2d_7 & f_1 &= 4d_8 & g_1 &= 2d_4 - 2d_7 \\ e_2 &= d_1 - d_2 - d_5 + d_9 & f_2 &= 2d_3 - 2d_6 & g_2 &= -d_1 - d_2 + d_5 + d_9 \end{aligned}$$

The second quadratic is

$$(k_2 t_5^2 + k_1 t_5 + k_0) t_6^2 + (j_2 t_5^2 + j_1 t_5 + j_0) t_6 + (i_2 t_5^2 + i_1 t_5 + i_0) = 0 \quad (5.9)$$

where

$$i_0 = h_1 + h_2 + h_5 + h_9 \qquad j_0 = 2h_4 + 2h_7 \qquad k_0 = h_1 - h_2 - h_5 + h_9$$
$$i_1 = 2h_3 + 2h_6 \qquad j_1 = 4h_8 \qquad k_1 = 2h_3 - 2h_6$$
$$i_2 = -h_1 + h_2 - h_5 + h_9 \qquad j_2 = 2h_4 - 2h_7 \qquad k_2 = -h_1 - h_2 + h_5 + h_9$$

The third quadratic is

$$(o_2 t_5^2 + o_1 t_5 + o_0) t_4^2 + (n_2 t_5^2 + n_1 t_5 + n_0) t_4 + (m_2 t_5^2 + m_1 t_5 + m_0) = 0 \qquad (5.10)$$

where

$$m_0 = l_1 + l_2 + l_5 + l_9 \qquad n_0 = 2l_4 + 2l_7 \qquad o_0 = l_1 - l_2 - l_5 + l_9$$
$$m_1 = 2l_3 + 2l_6 \qquad n_1 = 4l_8 \qquad o_1 = 2l_3 - 2l_6$$
$$m_2 = -l_1 + l_2 - l_5 + l_9 \qquad n_2 = 2l_4 - 2l_7 \qquad o_2 = -l_1 - l_2 + l_5 + l_9$$

The variable t_4 in equations (5.8) and (5.10) is eliminated using Sylvester's Dialytic method of elimination (c.f. Burnside and Panton [1889]). A 4th order polynomial is then obtained in terms of the variables t_5 and t_6

$$\begin{vmatrix} g_2 t_5^2 + g_1 t_5 + g_0 & f_2 t_5^2 + f_1 t_5 + f_0 & e_2 t_5^2 + e_1 t_5 + e_0 & 0 \\ 0 & g_2 t_5^2 + g_1 t_5 + g_0 & f_2 t_5^2 + f_1 t_5 + f_0 & e_2 t_5^2 + e_1 t_5 + e_0 \\ o_2 t_6^2 + o_1 t_6 + o_6 & n_2 t_6^2 + n_1 t_6 + n_6 & m_2 t_6^2 + m_1 t_6 + m_6 & 0 \\ 0 & o_2 t_6^2 + o_1 t_6 + o_6 & n_2 t_6^2 + n_1 t_6 + n_6 & m_2 t_6^2 + m_1 t_6 + m_6 \end{vmatrix} = 0.$$

After expanding and simplifying, the resultant in terms of t_5 and t_6 is given by

$$v_4 t_6^4 + v_3 t_6^3 + v_2 t_6^2 + v_1 t_6 + v_0 = 0 \qquad (5.11)$$

where

$$v_0 = p_4 t_5^4 + p_3 t_5^3 + p_2 t_5^2 + p_1 t_5 + p_0,$$
$$v_1 = q_4 t_5^4 + q_3 t_5^3 + q_2 t_5^2 + q_1 t_5 + q_0,$$
$$v_2 = r_4 t_5^4 + r_3 t_5^3 + r_2 t_5^2 + r_1 t_5 + r_0,$$
$$v_3 = s_4 t_5^4 + s_3 t_5^3 + s_2 t_5^2 + s_1 t_5 + s_0,$$
$$v_4 = u_4 t_5^4 + u_3 t_5^3 + u_2 t_5^2 + u_1 t_5 + u_0,$$

and p, q, r, and u are constants in terms of e, f, g, m, n and o.

Now equation (5.9) is rewritten as

$$w_2 t_6^2 + w_1 t_6 + w_0 = 0 \qquad (5.12)$$

where

$$w_0 = i_2 t_5^2 + i_1 t_5 + i_0,$$
$$w_1 = j_2 t_5^2 + j_1 t_5 + j_0,$$
$$w_2 = k_2 t_5^2 + k_1 t_5 + k_0.$$

Again using Sylvester's Dialytic method to eliminate the variable t_6 from equations (5.11) and (5.12) the combined matrix can be obtained as shown below.

$$\begin{vmatrix} w_2 & w_1 & w_0 & 0 & 0 & 0 \\ 0 & w_2 & w_1 & w_0 & 0 & 0 \\ 0 & 0 & w_2 & w_1 & w_0 & 0 \\ 0 & 0 & 0 & w_2 & w_1 & w_0 \\ v_4 & v_3 & v_2 & v_1 & v_0 & 0 \\ 0 & v_4 & v_3 & v_2 & v_1 & v_0 \end{vmatrix} = 0.$$

After some further algebraic manipulation a 16th order polynomial is obtained in terms of the variable t_5.

$$\begin{aligned} &A_{16}t_5^{16} + A_{15}t_5^{15} + A_{14}t_5^{14} + A_{13}t_5^{13} + A_{12}t_5^{12} + A_{11}t_5^{11} + A_{10}t_5^{10} + A_9t_5^9 \\ &+ A_8t_5^8 + A_7t_5^7 + A_6t_5^6 + A_5t_5^5 + A_4t_5^4 + A_3t_5^3 + A_2t_5^2 + A_1t_5 + A_0 = 0 \end{aligned} \tag{5.13}$$

where the A_i are functions of the fixed geometry only.

The maximum order of $tan(\beta_5/2)$ in the above equation is 16, and in the general case there will be 16 different solutions to the above equation. Since each value of $t_5 = tan(\beta_5/2)$, β_5 is uniquely determined in the interval $0 \leq \beta_5 < 360^0$. Thus β_4 and β_6, can now be computed by substituting for β_5 into equations (5.5) and (5.6). After substitution of β_5, each of these equations has the form

$$P\cos\beta_{4,6} + Q\sin\beta_{4,6} + R = 0 \tag{5.14}$$

yielding two solutions given by

$$\beta_{4,6} = 2\tan^{-1}\left(\frac{-Q \pm \sqrt{P^2 + Q^2 - R^2}}{R - P}\right). \tag{5.15}$$

The substitution of the solutions for β_4 and β_6 into eqn (5.7) will resolve the sign ambiguity since only one of the four possible combinations of solutions will satisfy the equation. It must also be determined whether any extraneous roots arise due to the process of elimination. That is, it must be determined if it is possible to obtain a univariate polynomial equation of lower degree. This can be done numerically by substituting each of the 16 values for β_5 into equations (5.5) and (5.6) to determine two values each of β_4 and β_6. Each of the four possible combinations of β_4 and β_6 can then be substituted into equation (5.7). There will be small numerical errors in the substitutions but these are comparatively small; thus the elimination process does not generate extraneous roots. A numerical example is given later to illustrate this. Note also that the number of real solutions depends on the input parameters of the problem.

5.2.2 Solution when the revolute joint axes on the platform are co-planar

If the revolute joint axes y_m in the platform are coplanar the following further relationships can be introduced

$$n_{m_z} = 0, \quad s_{m_z} = 0, \quad a_{m_x} = 0, \quad a_{m_y} = 0, \quad a_{m_z} = 1$$

thus reducing the closure equations to

$$d_3 s\beta_4 + d_4 s\beta_5 + d_5 c\beta_4 c\beta_5 + d_8 s\beta_4 s\beta_5 + d_9 = 0,$$
$$h_3 s\beta_5 + h_4 s\beta_6 + h_5 c\beta_5 c\beta_6 + h_8 s\beta_5 s\beta_6 + h_9 = 0,$$
$$l_3 s\beta_6 + l_4 s\beta_4 + l_5 c\beta_6 c\beta_4 + l_8 s\beta_6 s\beta_4 + l_9 = 0.$$

After carrying out the same elimination procedure, a 16th order polynomial is obtained. Those familiar with this solution procedure applied to Stewart platforms should note that the polynomial will not be in terms of even powers of t since the 0° position of the arms is along the z axes, not the x axes as is common in most Stewart platform solutions. The resulting angles β will still of course be mirrored through the plane formed by the y (or revolute joint) axes.

5.2.3 Solution when there is symmetry through the homokinetic plane

If there is symmetry through the plane formed by the points B_1, B_2 and B_3, then the upper arm angles β_4, β_5 and β_6 take the same value as the lower arm angles β_1, β_2 and β_3 respectively. The point O_p can be found by projecting the vector $\boldsymbol{B}_1$ on to the direction vector $\boldsymbol{w}_1$ (normal to plane of symmetry) and then multiplying by 2. Thus the displacement solution becomes

$$\boldsymbol{O}_p = 2\,\boldsymbol{B}_1^T\,\hat{\boldsymbol{w}}_1\,\hat{\boldsymbol{w}}_1.$$

Platform position and orientation relative to the base

The platform position and orientation relative to the base can now be found, i.e. the transformation matrix ${}^{0}\mathbf{T}_{u_2}$ that provides a relationship between the coordinate system $\mathcal{F}_{u_2v_2w_2}$ fixed to the platform, and the principal coordinate system $\mathcal{F}_0$ fixed to the base. The first step is to formulate an intermediate transformation matrix ${}^{u_1}\mathbf{T}_{u_2}$.

The vector defining the $\boldsymbol{u}_1$ axis expressed in the $\mathcal{F}_{u_2v_2w_2}$ system is

$${}^{u_2}\boldsymbol{u}_1 = \frac{{}^{u_2}\boldsymbol{B}_1 - {}^{u_2}\boldsymbol{B}_3}{\left|{}^{u_2}\boldsymbol{B}_1 - {}^{u_2}\boldsymbol{B}_3\right|}.$$

The $\boldsymbol{w}_1$ axis is the normal to the plane defined by the points B_1, B_2 and B_3 and so can be found by the vector cross product of $\overline{B_1B_2}$ and $\overline{B_1B_3}$.

$${}^{u_2}\boldsymbol{w}_1 = \frac{{}^{u_2}\overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times {}^{u_2}\overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\left|{}^{u_2}\overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times {}^{u_2}\overline{\boldsymbol{B}_1\boldsymbol{B}_3}\right|}.$$

The $\boldsymbol{v}_1$ axis is given by

$${}^{u_2}\boldsymbol{v}_1 = {}^{u_2}\boldsymbol{w}_1 \times {}^{u_2}\boldsymbol{u}_1.$$

The transformation matrix ${}^{u_1}\mathbf{T}_{u_2}$ is therefore

$${}^{u_1}\mathbf{T}_{u_2} = \begin{bmatrix} {}^{u_2}u_1(1) & {}^{u_2}v_1(1) & {}^{u_2}w_1(1) & {}^{u_2}B_1(1) \\ {}^{u_2}u_1(2) & {}^{u_2}v_1(2) & {}^{u_2}w_1(2) & {}^{u_2}B_1(2) \\ {}^{u_2}u_1(3) & {}^{u_2}v_1(3) & {}^{u_2}w_1(3) & {}^{u_2}B_1(3) \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The transformation matrix ${}^{u_2}\mathrm{T}_{u_1}$ is the inverse of the matrix ${}^{u_1}\mathrm{T}_{u_2}$ and can be easily found by noting that in general, for homogeneous transformation matrices that if

$$\mathrm{T} = \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

then in general

$$\mathrm{T}^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\boldsymbol{n}^T\boldsymbol{P} \\ s_x & s_y & s_z & -\boldsymbol{s}^T\boldsymbol{P} \\ a_x & a_y & a_z & -\boldsymbol{a}^T\boldsymbol{P} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Another intermediate transformation matrix ${}^{0}\mathrm{T}_{u_1}$ can be formulated as follows. The vector defining the direction of the $\boldsymbol{u}_1$ axis expressed in the $\mathscr{F}_0$ system is

$$\boldsymbol{u}_1 = \frac{\boldsymbol{B}_1 - \boldsymbol{B}_3}{\left|\boldsymbol{B}_1 - \boldsymbol{B}_3\right|}.$$

The $\boldsymbol{w}_1$ axis is the normal to the plane defined by the points B_1, B_2 and B_3 and so can be found by the vector cross product of $\overline{\boldsymbol{B}_1\boldsymbol{B}_2}$ and $\overline{\boldsymbol{B}_1\boldsymbol{B}_3}$, i.e.

$$\boldsymbol{w}_1 = \frac{\overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\left|\overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3}\right|}.$$

The $\boldsymbol{v}_1$ axis is given by

$$\boldsymbol{v}_1 = \boldsymbol{w}_1 \times \boldsymbol{u}_1.$$

The transformation matrix ${}^{0}\mathrm{T}_{u_1}$ is therefore

$${}^{0}\mathrm{T}_{u_1} = \begin{bmatrix} u_1(1) & v_1(1) & w_1(1) & B_1(1) \\ u_1(2) & v_1(2) & w_1(2) & B_1(2) \\ u_1(3) & v_1(3) & w_1(3) & B_1(3) \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The transformation matrix ${}^{0}\mathrm{T}_{u_2}$ is found by multiplying the intermediate transformation matrices together

$${}^{0}\mathrm{T}_{u_2} = {}^{0}\mathrm{T}_{u_1}\,{}^{u_1}\mathrm{T}_{u_2}.$$

The displacement vector $\boldsymbol{O}_p$ is given by the first three elements of the 4th column of ${}^{0}\mathrm{T}_{u_2}$

$$\boldsymbol{O}_p = \begin{pmatrix} {}^{0}\mathrm{T}_{u_2}(1,4) \\ {}^{0}\mathrm{T}_{u_2}(2,4) \\ {}^{0}\mathrm{T}_{u_2}(3,4) \end{pmatrix}.$$

Calculation of the orientation of the platform

The orientation of the platform can be calculated by equating the 1st 3×3 partition of the matrix

${}^{0}\mathbf{T}_{u_2}$ with a 3-2-3 Euler rotation matrix and the three Euler angles gleaned by comparing entries of both matrices. For beam aiming applications rotation about the 'line of sight' has no effect on circularly polarised electromagnetic waves so only the two angles θ_{3a} and θ_2 (c.f. figure 5.4) need to be resolved. Note, it is assumed that the object being targeted is a large distance from the platform, thus the small angular difference in measuring the tracked object from the frame $\mathcal{F}_0$ as compared to measuring the tracked object from the platform frame $\mathcal{F}_{u_2v_2w_2}$ is ignored, i.e for the purposes of this exercise the two frames are assumed to be coincident.

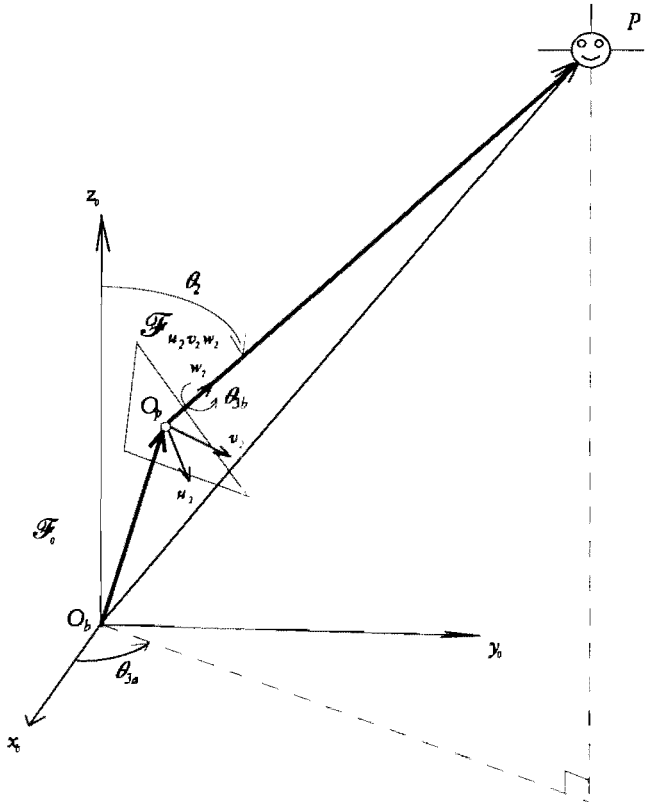


Figure 5.4 Beam aiming angles.

The 3-2-3 Euler rotation matrix is given by

$$\begin{bmatrix} c\theta_{3b}c\theta_2c\theta_{3a} - s\theta_{3b}s\theta_{3a} & -s\theta_{3b}c\theta_2c\theta_{3a} - c\theta_{3b}s\theta_{3a} & s\theta_2c\theta_{3a} \\ c\theta_{3b}c\theta_2s\theta_{3a} + s\theta_{3b}c\theta_{3a} & -s\theta_{3b}c\theta_2s\theta_{3a} + c\theta_{3b}c\theta_{3a} & s\theta_2s\theta_{3a} \\ -c\theta_{3b}s\theta_2 & s\theta_{3b}s\theta_2 & c\theta_2 \end{bmatrix}.$$

Equating this rotation matrix to the 1st 3×3 partition of the matrix ${}^{0}\mathbf{T}_{u_2}$ the aiming angles θ can be found. The angle θ_2 is given by

$$\theta_2 = \text{acos}\left({}^{0}\text{T}_{u_2}(3,3)\right) \qquad 0^0 \le \theta_2 \le 180^0. \tag{5.16}$$

The angle θ_{3a} is given by

$$\theta_{3a} = \text{atan2}\left({}^{0}\text{T}_{u_2}(2,3),\ {}^{0}\text{T}_{u_2}(1,3)\right) \tag{5.17}$$

and the redundant angle θ_{3b} is given by

$$\theta_{3b} = \text{atan2}\left({}^{0}\text{T}_{u_2}(3,2),\ -{}^{0}\text{T}_{u_2}(3,1)\right). \tag{5.18}$$

Note, if the mechanism has symmetry through homokinetic plane defined by the points B_1, B_2 and B_3, then the redundant angle $\theta_{3b} = -\theta_{3a}$.

An alternative method for finding the pointing angles θ_{3a} and θ_2 can be found if the mechanism has symmetry through the homokinetic plane defined by the points B_1, B_2 and B_3.

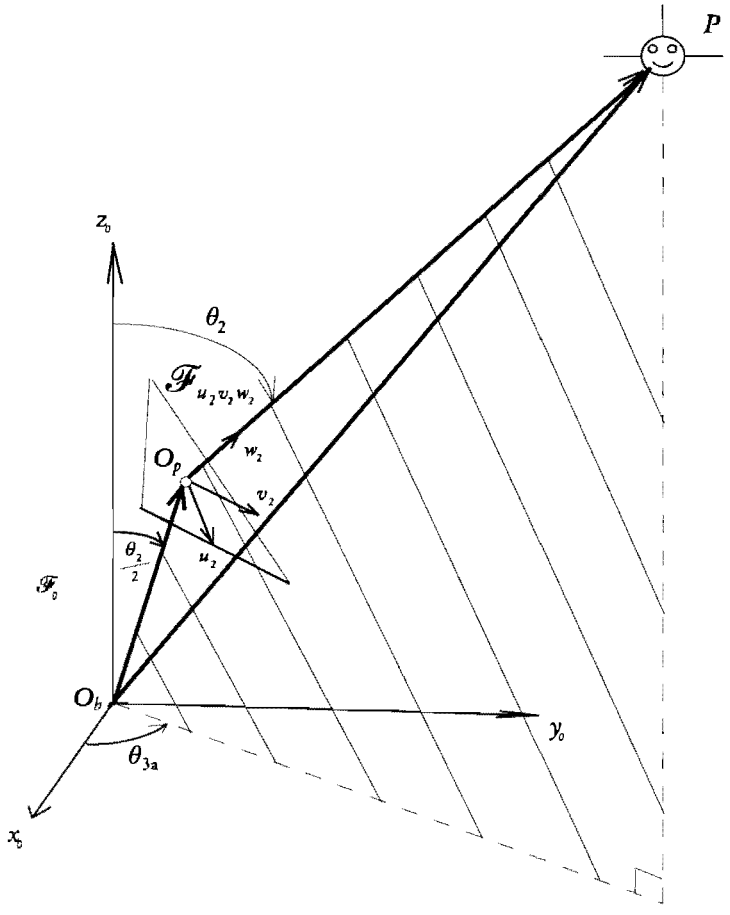


Figure 5.5 Beam aiming angles for a symmetrical mechanism.

The displacement vector $\boldsymbol{O_p}$ will then have the same rotation angle θ_{3a} about the axis z_0 as the beam aiming direction vector $\overline{\boldsymbol{O_pP}}$ (c.f. figure 5.5). The angle between the axis z_0 and $\overline{\boldsymbol{O_bO_p}}$ will be equal to the angle between $\overline{\boldsymbol{O_bO_p}}$ and $\overline{\boldsymbol{O_pP}}$. Thus the position of the platform $\boldsymbol{O_p}$ can be written as

$$\boldsymbol{O_p} = \begin{pmatrix} |\boldsymbol{O_p}| \sin\frac{\theta_2}{2}\cos\theta_{3a} \\ |\boldsymbol{O_p}| \sin\frac{\theta_2}{2}\sin\theta_{3a} \\ |\boldsymbol{O_p}| \cos\frac{\theta_2}{2} \end{pmatrix} \tag{5.19}$$

giving

$$\theta_{3a} = \text{atan2}\left(O_{p(2)}, O_{p(1)}\right)$$

and

$$\theta_2 = 2\cos^{-1}\left(\frac{O_{p(3)}}{|O_p|}\right) \qquad 0^0 \le \theta_2 \le 180^0 .$$

Note, this alternative approach was carried out to facilitate formulation of the Jacobian matrix derived in Appendix C.

Numerical example for the forward kinematics

A numerical example is computed to demonstrate the multiple solutions/configurations for a set of driven arm angles. The direct kinematic equations yield sixteen possible platform positions and as shown in the numerical study below, not all of these are physically realised, i.e. many of the roots occur in complex conjugate pairs. Thus in general there are always at least two possible solutions.

Some of the parameters defining the mechanism's geometry are shown below. The base and platform are equilateral triangles with revolute joints positioned at each side midpoint and are 280.33mm from the centroids O_b and O_p. The arms are 841.00mm long except for l_4 which is 700.00mm long.

The three driven arm angles are

$$\beta_1 = 60°,\ \beta_2 = 45°,\ \beta_3 = 30°.$$

The three spherical joints (expressed in the $\mathcal{F}_0$ coordinate system) are at positions

$$\boldsymbol{B}_1 = \begin{pmatrix} 1.008657 \\ 0 \\ 0.420500 \end{pmatrix},\ \boldsymbol{B}_2 = \begin{pmatrix} -0.437503 \\ 0.7577781 \\ 0.5946768 \end{pmatrix},\ \boldsymbol{B}_3 = \begin{pmatrix} -0.350415 \\ -0.606937 \\ 0.7283274 \end{pmatrix}.$$

The three passive revolute joints (expressed in the $\mathcal{F}_{u_2v_2w_2}$ coordinate system) are at positions

$${}^{u_2}\boldsymbol{P}_4 = \begin{pmatrix} 0.280330 \\ 0 \\ 0 \end{pmatrix},\ {}^{u_2}\boldsymbol{P}_5 = \begin{pmatrix} -0.1401650 \\ 0.2427729 \\ 0 \end{pmatrix},\ {}^{u_2}\boldsymbol{P}_6 = \begin{pmatrix} -0.1401650 \\ -0.2427729 \\ 0 \end{pmatrix}.$$

The three spherical joints (expressed in the $\mathcal{F}_{u_2v_2w_2}$ coordinate system) are at positions

$${}^{u_2}\boldsymbol{B}_1 = \begin{pmatrix} 0.28033 + 0.7\ \sin\beta_4 \\ 0 \\ -0.7\ \cos\beta_4 \end{pmatrix},\ {}^{u_2}\boldsymbol{B}_2 = \begin{pmatrix} -0.140165 - 0.4205\ \sin\beta_5 \\ 0.242747 + 0.728327\ \sin\beta_5 \\ -0.841\ \cos\beta_5 \end{pmatrix},\ {}^{u_2}\boldsymbol{B}_3 = \begin{pmatrix} 0.140165 - 0.4205\ \sin\beta_6 \\ -0.242747 - 0.728327\ \sin\beta_6 \\ -0.841\ \cos\beta_6 \end{pmatrix}.$$

For this example the polynomial equation in terms of t_5 is

$$\begin{aligned} &0.05701474105699 \times 10^5\ t_5^{16} - 0.10071233933817 \times 10^5\ t_5^{15} + 1.12838873402140 \times 10^5\ t_5^{14} \\ &- 1.55913343050819 \times 10^5\ t_5^{13} - 3.55391140088964 \times 10^5\ t_5^{12} - 0.93129250187841 \times 10^5\ t_5^{11} \\ &+ 6.11465029625884 \times 10^5\ t_5^{10} + 0.47707321944806 \times 10^5\ t_5^{9} - 3.21330445223874 \times 10^5\ t_5^{8} \\ &+ 0.47707321944805 \times 10^5\ t_5^{7} + 6.11465029625884 \times 10^5 t_5^{6} - 0.93129250187841 \times 10^5\ t_5^{5} \\ &- 3.55391140088964 \times 10^5\ t_5^{4} - 1.55913343050819 \times 10^5\ t_5^{3} + 1.12838873402140 \times 10^5\ t_5^{2} \\ &- 0.10071233933817 \times 10^5\ t_5^{1} + 0.05701474105699 \times 10^5 = 0. \end{aligned}$$

Of the 16 roots of the polynomial, only 4 are found to be real. These are used to find β_5, which is back substituted to find the angles β_4 and β_6. Thus the displacement and orientation of the platform relative to the base can be found. The four real solutions with corresponding mechanism configurations are shown in figures 5.6 a-d.

Configuration (a)

Upper arm angles

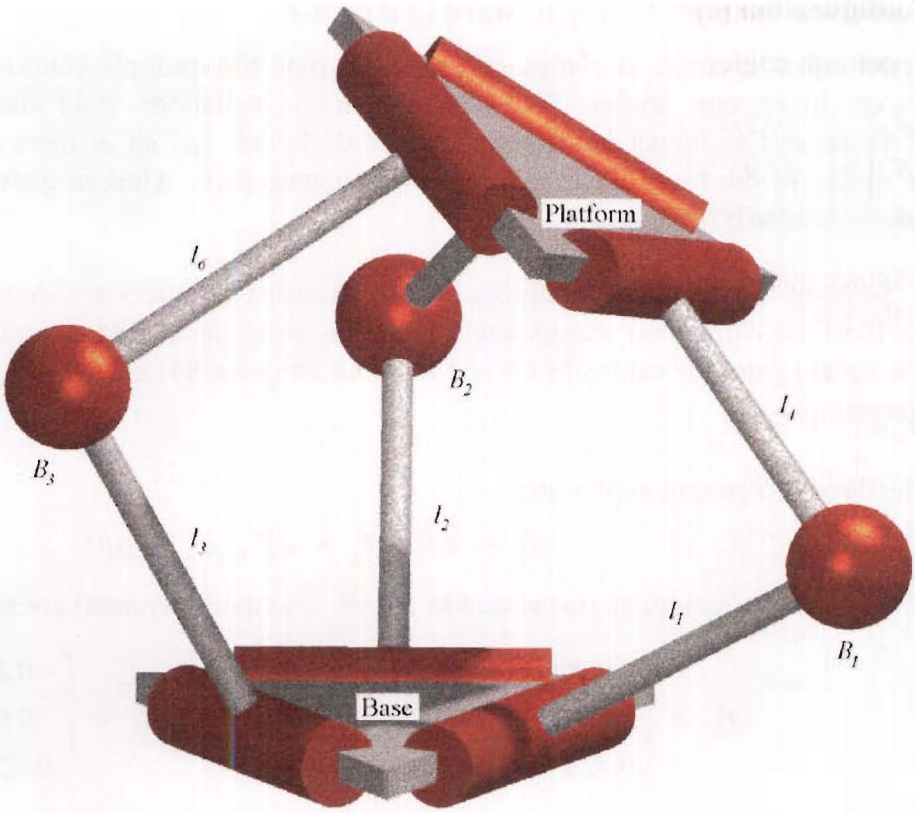
$$\begin{pmatrix}\beta_4\\ \beta_5\\ \beta_6\end{pmatrix}=\begin{pmatrix}106.2977^\circ\\ 133.3762^\circ\\ 151.7061^\circ\end{pmatrix}.$$

Displacement vector of the platform

$$\boldsymbol{O}_p=\begin{pmatrix}0.2743\\ 0.1122\\ 1.0477\end{pmatrix}.$$

Aiming angles of the platform

$$\begin{pmatrix}\theta_{3a}\\ \theta_2\end{pmatrix}=\begin{pmatrix}25.5601^\circ\\ 32.6710^\circ\end{pmatrix}.$$



Configuration a.

Configuration (b)

Upper arm angles

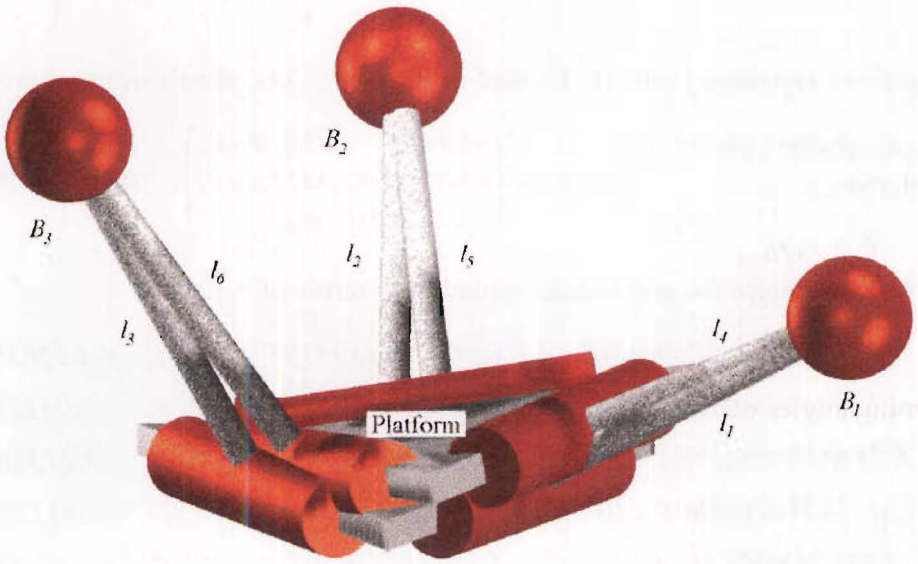
$$\begin{pmatrix}\beta_4\\ \beta_5\\ \beta_6\end{pmatrix}=\begin{pmatrix}73.7023^\circ\\ 46.6238^\circ\\ 28.2939^\circ\end{pmatrix}.$$

Displacement vector of the platform

$$\boldsymbol{O}_p=\begin{pmatrix}0.1009\\ 0.0056\\ 0.0723\end{pmatrix}.$$

Aiming angles of the platform

$$\begin{pmatrix}\theta_{3a}\\ \theta_2\end{pmatrix}=\begin{pmatrix}191.5569^\circ\\ 9.5267^\circ\end{pmatrix}.$$



Configuration b.

Configuration (c)

Upper arm angles

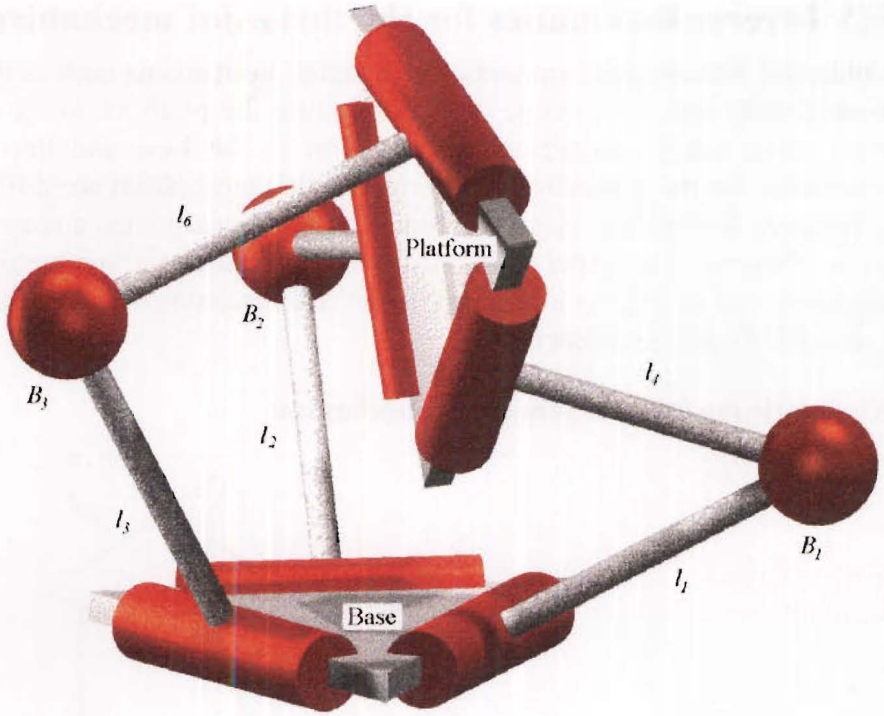
$$\begin{pmatrix}\beta_4\\ \beta_5\\ \beta_6\end{pmatrix}=\begin{pmatrix}30.2162^\circ\\ 87.9574^\circ\\ -158.5908^\circ\end{pmatrix}.$$

Displacement vector of the platform

$$\boldsymbol{O}_p=\begin{pmatrix}0.2413\\ -0.1034\\ 0.8286\end{pmatrix}.$$

Aiming angles of the platform

$$\begin{pmatrix}\theta_{3a}\\ \theta_2\end{pmatrix}=\begin{pmatrix}40.9978^\circ\\ 84.5017^\circ\end{pmatrix}.$$



Configuration c.

Configuration (d)

Upper arm angles

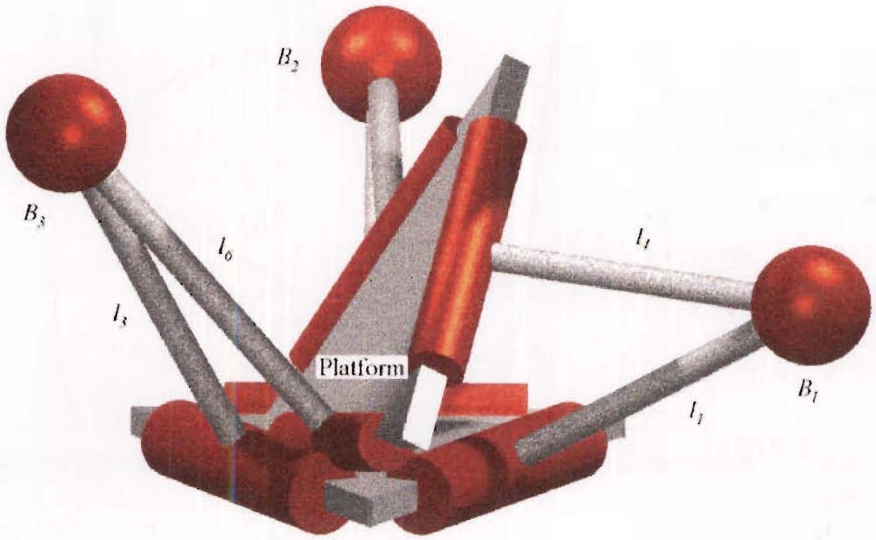
$$\begin{pmatrix}\beta_4\\ \beta_5\\ \beta_6\end{pmatrix}=\begin{pmatrix}149.7838^\circ\\ 92.0426^\circ\\ -21.4092^\circ\end{pmatrix}.$$

Displacement vector of the platform

$$\boldsymbol{O}_p=\begin{pmatrix}0.1526\\ -0.1579\\ 0.3294\end{pmatrix}.$$

Aiming angles of the platform

$$\begin{pmatrix}\theta_{3a}\\ \theta_2\end{pmatrix}=\begin{pmatrix}222.2819^\circ\\ 61.2853^\circ\end{pmatrix}.$$



Configuration d.

Figure 5.6 Mechanism configurations corresponding to the four real solutions.

It should be noted that if a mirror were to lie in the plane defined by the three points B_1, B_2 and B_3, then configuration a would be a mirror image of configuration b, and configuration c would be a mirror image of configuration d. Note also that configuration c has been slightly rotated so that the spherical joint B_2 is not obscured.

5.3 Inverse kinematics for the three dof mechanism with symmetry

Unlike the Stewart platform and other parallel mechanisms such as those described by Lee and Shah [1988] with only single links connecting the platform to the base, this mechanism has three serial chains connecting the platform to the base and thus the closed form inverse kinematics for the generalised geometry for this mechanism are difficult to solve. If symmetry is assumed through the plane defined by the spherical joints, a reasonably simple solution can to be obtained. Fortunately, for most real applications a symmetrical mechanism is usually desirable, and as long as tolerances on the real mechanism are kept tight, the following inverse kinematic model is satisfactory.

Kinematic model for a symmetric mechanism

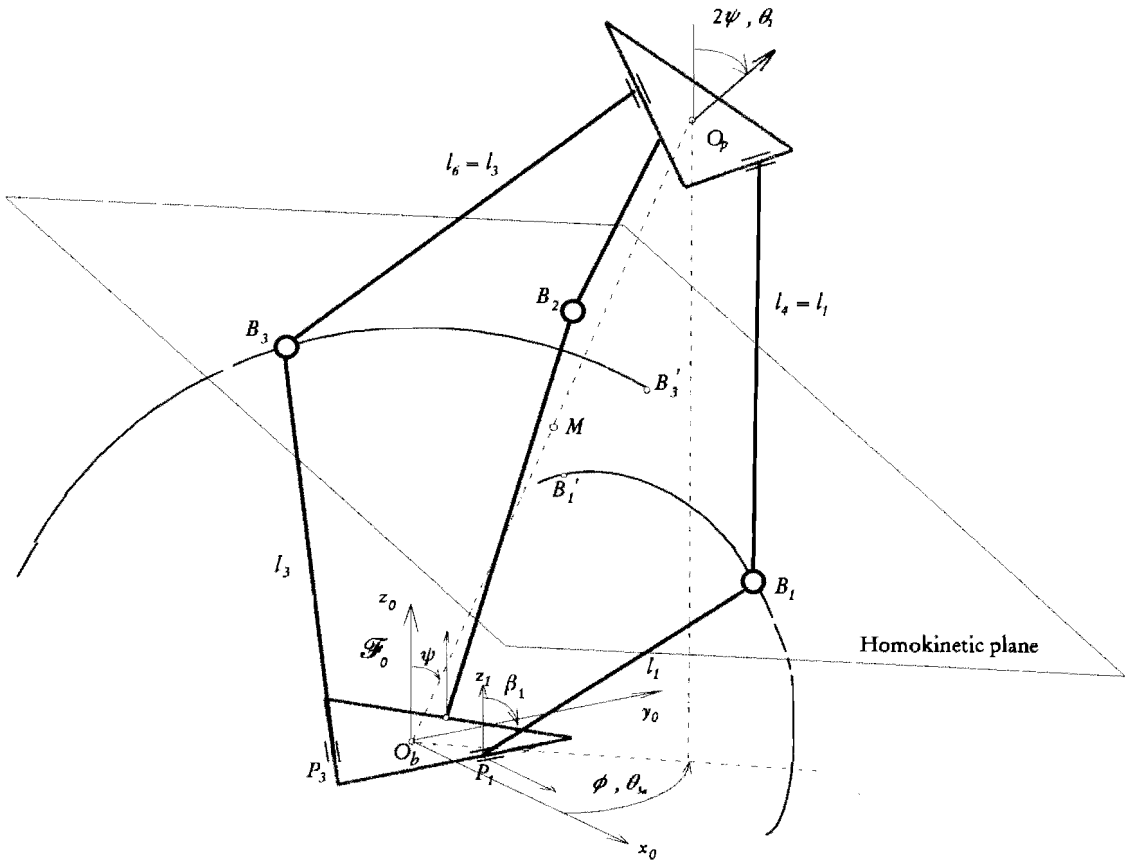


Figure 5.7 Kinematic model for inverse kinematic solution (some details have been omitted for clarity).

The driven arms attached to the base of the mechanism describe an arc as shown in figure 5.7. Generally the arcs must intersect the homokinetic plane at two points. These points represent the possible positions that the driven arms may assume for a given platform orientation and position. The method described in Appendix B for finding the intersection points of a circle and a plane, is applied to find the points B_n and the unused alternative points B_n'. Note that the mechanism must be deliberately moved through a singularity in order to place the ball joints at B_n'.

The aiming/pointing coordinates of a satellite and hence the platform may be specified by only two angular parameters θ_{3a}, θ_2. Since this mechanism has three dof, a third redundant freedom $|\boldsymbol{O}_p|$ (the distance between the base O_b and platform O_p) must also be specified. Given the two pointing angles θ_{3a} and θ_2 of the platform and a specified distance between the platform and base $|\boldsymbol{O}_p|$, the position of the platform relative to the base can be written as

$$\boldsymbol{O}_p = \begin{pmatrix} |\boldsymbol{O}_p| \sin\frac{\theta_2}{2}\cos\theta_{3a} \\ |\boldsymbol{O}_p| \sin\frac{\theta_2}{2}\sin\theta_{3a} \\ |\boldsymbol{O}_p| \cos\frac{\theta_2}{2} \end{pmatrix}.$$

$\boldsymbol{O}_p$ is the normal to the homokinetic plane (due to symmetry through the plane) thus defining the orientation of the plane. The position of the plane is such that it bisects $\boldsymbol{O}_p$ at M, i.e. M is midway between O_b and O_p

$$\boldsymbol{M} = \tfrac{1}{2}\boldsymbol{O}_p.$$

$\boldsymbol{P}_n$ are the positions of the centre of the circles describing the paths made by the end points of the arms and are given by

$$\boldsymbol{P}_1 = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix}, \quad \boldsymbol{P}_2 = \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix}, \quad \boldsymbol{P}_3 = \begin{pmatrix} a_3 \\ b_3 \\ c_3 \end{pmatrix}.$$

The $\boldsymbol{y}_n$ axes are the revolute axes of the driven arms and so define the orientation of the axes (circles). Their directions are given by

$$\boldsymbol{y}_1 = \begin{pmatrix} s_{1x} \\ s_{1y} \\ s_{1z} \end{pmatrix}, \quad \boldsymbol{y}_2 = \begin{pmatrix} s_{2x} \\ s_{2y} \\ s_{2z} \end{pmatrix}, \quad \boldsymbol{y}_3 = \begin{pmatrix} s_{3x} \\ s_{3y} \\ s_{3z} \end{pmatrix}.$$

The length of the arms (radii of the circles) are l_n.

The points S_n (see Appendix B for a detailed explanation of S_n) which are the points midway between the intersection points B_n and B_n' can now be found. These points S_n are the intersection of the vectors $(\boldsymbol{y}_n \times \boldsymbol{O}_p) \times \boldsymbol{y}_n$ through P_n and the plane. The equation of the plane can be written as

$$\boldsymbol{O}_p^T \boldsymbol{S}_n - \boldsymbol{O}_p^T \boldsymbol{M} = 0. \tag{5.20}$$

The position vector $\boldsymbol{S}_n$ can be written in parametric form as

$$\boldsymbol{S}_n = \boldsymbol{P}_n + f_n\{(\boldsymbol{y}_n \times \boldsymbol{O}_p) \times \boldsymbol{y}_n\} \tag{5.21}$$

where f_n is a scalar quantity obtained by combining equations (5.20) and (5.21) and is given by

$$f_n = \frac{\boldsymbol{O}_p^T \boldsymbol{M} - \boldsymbol{O}_p^T \boldsymbol{P}_n}{\boldsymbol{O}_p^T\{(\boldsymbol{y}_n \times \boldsymbol{O}_p) \times \boldsymbol{y}_n\}}.$$

The vectors $\overline{SB_n}$ can be written as

$$\overline{SB_n} = t_n(y_n \times O_p)$$

where t_n is an unknown scalar quantity.

The position vectors $\boldsymbol{B_n}$, are the points of intersection between the plane and circle and are given by

$$B_n = S_n + \overline{SB_n}$$

The distance from the points P_n (centre of the circle) to the points B_n and B_n' (points of intersection with the homokinetic plane) must be length l_n. Hence

$$l_n = |B_n - P_n|$$

or

$$l_n = |S_n + t_n(y_n \times O_p) - P_n|. \qquad (5.22)$$

Squaring both sides of equation (5.22) and rearranging gives a quadratic in the unknown quantities t_n. The solution for t_n can be readily found using an algebraic manipulation package and is shown in Appendix B. The intersection points $\boldsymbol{B_n}$ are then given by

$$B_n = S_n + t_n(y_n \times O_p). \qquad (5.23)$$

5.3.1 Calculation of the arm angles

Two possible arm angles corresponding to the arm positions B_n and B_n' can be found for each arm. These can be found by using the dot product rule to find the angle between the z_n axes $(0,0,1)^T$ and the position vector for the arm end points $\boldsymbol{B_n}$ expressed in their respective $\mathcal{F}_n$ coordinate frames. The arm angles are given by

$$\beta_n = cos^{-1}\left(\frac{{}^n\mathbf{T}_0 B_{n(3)}}{|{}^n\mathbf{T}_0 B_n|}\right).$$

The inverse kinematic equations yield two possible positions for each arm which indicates that there are eight possible solutions to the inverse kinematic problem.

5.4 Direct kinematics for the two dof mechanism with symmetry

5.4.1 Introduction

The following model of the two dof mechanism applies only to the particular case where there is symmetry about the plane defined by the spherical points B_1, B_2 and B_3. The direct kinematics of the two dof mechanism differs from that of the three dof mechanism since the introduction of a strut between the points O_b and O_p means that only two independent arm angles need be specified in order to calculate the two beam aiming angles θ_{3a} and θ_2 determining the pointing direction of the platform.

5.4.2 Kinematic solution for the two dof mechanism with symmetry

The two dof mechanism shown in figure 5.8 consists of two actuated arms positioned at points P_1 and P_2 attached to the base and one passive arm at point P_3 attached to the base. Three upper passive arms are attached to the platform. Each of the three base arms is connected to its respective opposing platform arm via a spherical joint. A central strut is fixed to both platforms via one Hookes joint at point O_p and one spherical joint at point O_b.

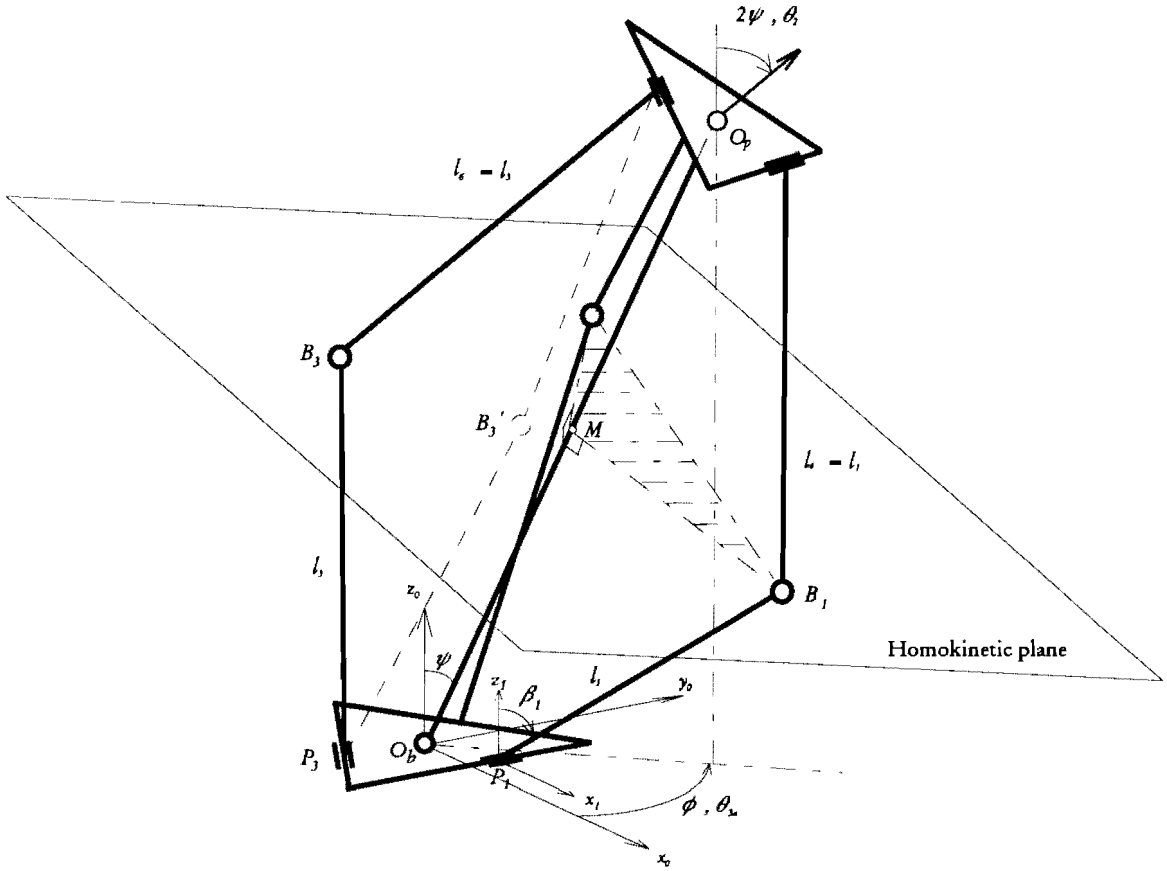


Figure 5.8 Schematic of two dof mechanism. Note the two possible positions that the third arm l_3 may assume.

Given the two arm angles β_1 and β_2, the position of the points B_1 and B_2 expressed in their respective local systems $\mathcal{F}_n$ can be written as

$$ {}^{n}\boldsymbol{B}_n = \begin{pmatrix} l_n \sin\beta_n \\ 0 \\ l_n \cos\beta_n \\ 1 \end{pmatrix}. $$

The points $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ can be expressed in the $\mathcal{F}_0$ frame by multiplying by the transformation matrices ${}^{0}\mathbf{T}_1, {}^{0}\mathbf{T}_2$

$$ \boldsymbol{B}_n = {}^{0}\mathbf{T}_n \, {}^{n}\boldsymbol{B}_n . $$

Due to symmetry, the point of intersection M between the homokinetic plane and the central strut must lie mid way between points O_b and O_p. By formulating the three constraint equations below the position of the point M can be found. The displacement vector $\boldsymbol{O}_p$ of the platform is then given by $2\,\boldsymbol{M}$.

The first constraint equation can be formulated by noting that the angle between the vectors $\overline{MO_b}$ and $\overline{MB_1}$ must be 90° for the point M to lie in the homokinetic plane, i.e.

$$\overline{MO_b}^T \overline{MB_1} = 0 \tag{5.24}$$

where $\overline{MO_b} = \boldsymbol{O}_b - \boldsymbol{M}$ and $\overline{MB_1} = \boldsymbol{B}_1 - \boldsymbol{M}$.

The second constraint equation can be formulated by noting angle between the vectors $\overline{MO_b}$ and $\overline{MB_2}$ must also be 90° for the point M to lie in the homokinetic plane, i.e.

$$\overline{MO_b}^T \overline{MB_2} = 0 \tag{5.25}$$

where $\overline{MB_2} = B_2 - M$.

The third constraint equation can be formulated by noting that the magnitude of the position vector $\boldsymbol{M}$ is half the magnitude of the position vector $\boldsymbol{O}_p$, i.e.

$$|M| = \tfrac{1}{2}|O_p|. \tag{5.26}$$

Equation (5.24) can be written as

$$B_{1(1)}M_{(1)} + B_{1(2)}M_{(2)} + B_{1(3)}M_{(3)} - (M_{(1)}^2 + M_{(2)}^2 + M_{(3)}^2) = 0. \tag{5.27}$$

Equation (5.25) can be written as

$$B_{2(1)}M_{(1)} + B_{2(2)}M_{(2)} + B_{2(3)}M_{(3)} - (M_{(1)}^2 + M_{(2)}^2 + M_{(3)}^2) = 0. \tag{5.28}$$

Squaring both sides of equation (5.26) gives

$$M_{(1)}^2 + M_{(2)}^2 + M_{(3)}^2 = \tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2. \tag{5.29}$$

Substituting for $M_{(1)}^2 + M_{(2)}^2 + M_{(3)}^2$ into equations (5.27) and (5.28) leads to

$$B_{1(1)}M_{(1)} + B_{1(2)}M_{(2)} + B_{1(3)}M_{(3)} - \tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2 = 0, \tag{5.30}$$

$$B_{2(1)}M_{(1)} + B_{2(2)}M_{(2)} + B_{2(3)}M_{(3)} - \tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2 = 0. \tag{5.31}$$

From equation (5.30) $M_{(1)}$ can be written in terms of $M_{(2)}$ and $M_{(3)}$

$$M_{(1)} = \frac{\tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2 - B_{1(2)}M_{(2)} - B_{1(3)}M_{(3)}}{B_{1(1)}}. \tag{5.32}$$

Substituting for $M_{(1)}$ into equation (5.31) gives

$$a\,M_{(2)} + b\,M_{(3)} + c = 0 \tag{5.33}$$

where

$$a = \frac{-B_{1(2)}B_{2(1)}}{B_{1(1)}} + B_{2(2)}, \qquad b = \frac{-B_{1(3)}B_{2(1)}}{B_{1(1)}} + B_{2(3)}, \qquad c = \frac{\tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2 B_{2(1)}}{B_{1(1)}} - \tfrac{1}{4}\left|\boldsymbol{O}_p\right|^2.$$

Substituting for $M_{(1)}$ into equation (5.29) gives

$$d\,M_{(2)}^2 + e\,M_{(3)}^2 + f M_{(2)} + g\,M_{(3)} + hM_{(2)}M_{(3)} + i = 0 \tag{5.34}$$

where

$$d = 1 + \frac{B_{1(2)}^2}{B_{1(1)}^2}, \qquad e = 1 + \frac{B_{1(3)}^2}{B_{1(1)}^2}, \qquad f = \frac{-\frac{1}{2}B_{1(2)}\left|\boldsymbol{O}_p\right|^2}{B_{1(1)}^2},$$

$$g = \frac{-\frac{1}{2}B_{1(3)}\left|\boldsymbol{O}_p\right|^2}{B_{1(1)}^2}, \qquad h = \frac{2B_{1(2)}B_{1(3)}}{B_{1(1)}^2}, \qquad i = \frac{(\frac{1}{4})^2\left|\boldsymbol{O}_p\right|^4}{B_{1(1)}^2} - \frac{1}{4}\left|\boldsymbol{O}_p\right|^2.$$

Equation (5.33) can be rewritten as

$$M_{(2)} = \frac{-b\,M_{(3)} - c}{a} \tag{5.35}$$

and substitute for $M_{(2)}$ into equation (5.34) to get

$$j\,M_{(3)}^2 + k\,M_{(3)} + l = 0 \tag{5.36}$$

where

$$j = \frac{b^2 d}{a^2} + e - \frac{bh}{a}, \qquad k = \frac{2bcd}{a^2} - \frac{bf}{a} + g - \frac{ch}{a}, \qquad l = \frac{c^2 d}{a^2} - \frac{cf}{a} + i.$$

The third component of the position vector $\boldsymbol{M}$ can be found by solving the quadratic equation (5.36) to get

$$M_{(3)} = \frac{-k \pm \sqrt{k^2 - 4jl}}{2j}. \tag{5.37}$$

The first and second components of the position vector $\boldsymbol{M}$ can be found by back substitution into equations (5.32) and (5.35). Note that the two possible solutions to equation (5.37) are due to the two positions that the third arm attached to the base may assume.

Because of symmetry about the homokinetic plane, the position vector $\boldsymbol{O}_p$ of the top platform is given by

$$\boldsymbol{O}_p = 2\boldsymbol{M}.$$

The pointing direction of the top platform can be calculated as for the three dof mechanism (c.f. section 5.3).

5.5 Inverse kinematics for the two dof mechanism with symmetry

The geometric model is the same as for the direct case shown in figure 5.8. Again the model of the two dof mechanism applies only to the particular case where there is symmetry about the plane defined by the spherical points B_1, B_2 and B_3. The inverse kinematics of the two dof mechanism differs from that of the three dof mechanism since the introduction of a strut between the points O_b and O_p means that only the two platform pointing angles θ_{3a} and θ_2 need be specified in order to determine the geometry of the mechanism, i.e. there is not a third redundant plunging freedom $|\boldsymbol{O}_p|$ as with the three dof mechanism. The calculation of the actuated arm angles β_1 and β_2 is the same as for the three dof mechanism in section 5.3.

Chapter 6

Static analysis

6.1 Introduction

In this chapter a static analysis of the mechanisms was performed so that an appreciation of the gravity loading on the robot can be gained. The results from this static analysis can then be compared with experimental values that were collected by taking measurements using a spring balance. In Chapter 9 a rigid body dynamic analysis is carried out and although the same information is gained from the dynamic analysis, this analysis presented here was done at a much earlier stage. It is also invaluable in providing a check on the static part of the dynamic analysis.

6.2 Static model of the three dof mechanism

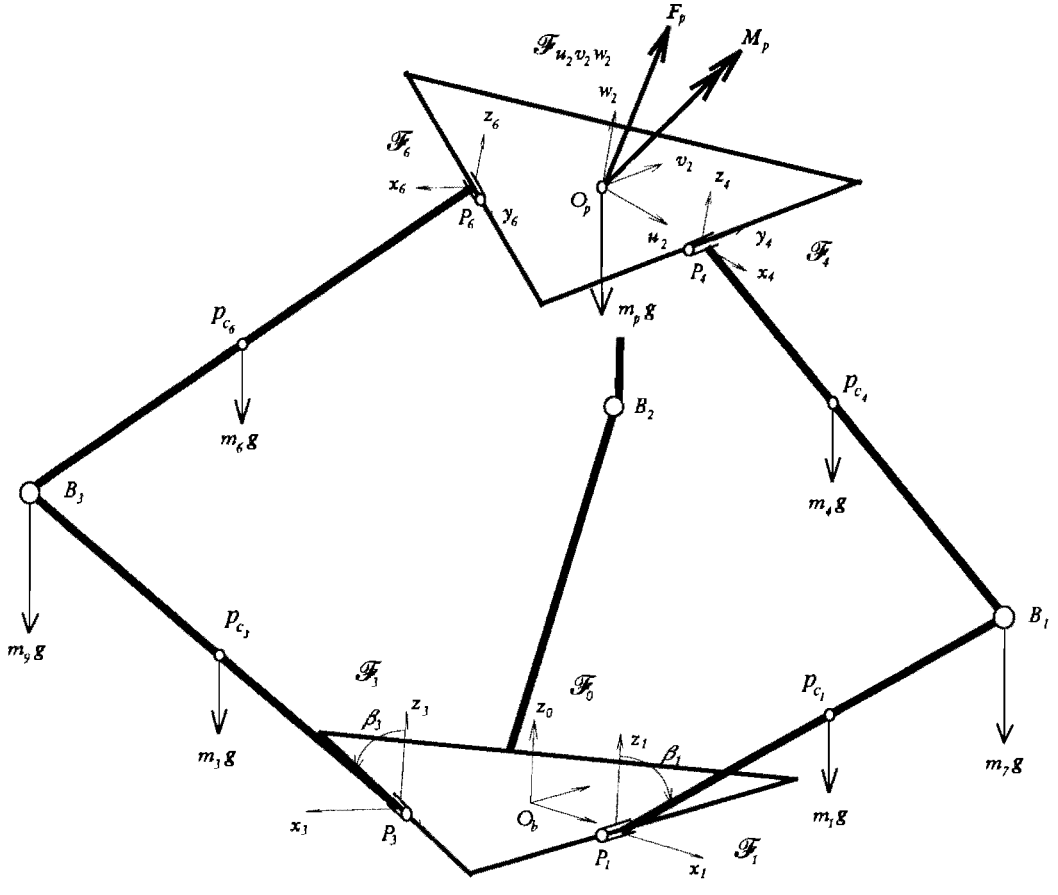


Figure 6.1 Model showing external forces acting on three dof mechanism.

For the following static analysis the mechanism is assumed to be symmetrical about the homokinetic plane defined by the points B_1, B_2 and B_3, and each triplet of revolute joint axes y is assumed to be coplanar. Although the analysis could be extended to a completely general

mechanism, in the interests of computational difficulty only the particular case outlined above is modeled.

The arms are modelled as lumped masses with their mass centres at points p_{c_i} (i = 1-6) as shown in figure 6.1. The spherical joints are modelled as lumped masses with their mass centres located points B_n (n = 1,2,3). The platform is modelled as a lumped mass with its centroid located at point O_p. A known external force $\boldsymbol{F}_P$ and a moment $\boldsymbol{M}_P$ acting through and about the point O_p can be applied.

6.3 Formulation of the static equations of equilibrium

Given that the three lower arms are fixed in position, if an imaginary cut is made at the three spherical joints B_1, B_2 and B_3, the upper portion of the mechanism will form a statically determinant structure supported at the three spherical joints. The three spherical joint force reaction vectors $\boldsymbol{N}_1$, $\boldsymbol{N}_2$ and $\boldsymbol{N}_3$ can be split into three independent orthogonal components; therefore there will be 9 unknown reaction components. Three scalar equations are formulated in terms of the reactions $\boldsymbol{N}$ owing to the fact that moments cannot be sustained about the platform revolute joint y_m axes (m = 4,5,6). Six more independent equations can be formulated from the 6 equations of static equilibrium, i.e. summing the forces and moments and equating to zero. Thus there will be 9 equations in terms of the 9 unknown spherical joint reaction vector components.

6.3.1 Summing the moments about the platform revolute joints

First formulate three scalar equations in terms of the reactions $\boldsymbol{N}$ by summing the moments about the platform revolute joints and equating them to zero. Consider the forces acting on the upper arm 4; the forces of interest are shown below in figure 6.2. The two forces of interest are the reaction vector $\boldsymbol{N}_1$ and the gravity force $m_4\boldsymbol{g}$ due to the mass of the passive arm.

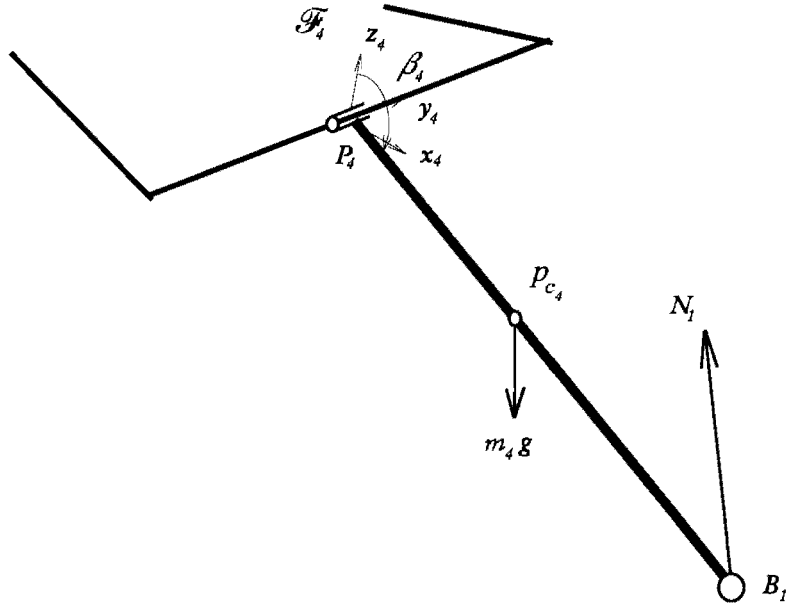


Figure 6.2 Forces acting on the upper arm 4.

The moments due to the weight of the arms and reaction forces are given by

$$ {}^{m}\boldsymbol{M}_{P_m} = {}^{m}\overline{\boldsymbol{P}_m\boldsymbol{p}_{c_m}} \times m_m {}^{m}\boldsymbol{g} + {}^{m}\overline{\boldsymbol{P}_m\boldsymbol{B}_n} \times {}^{m}\boldsymbol{N}_n \qquad \text{n= 1,2,3 and m = n+3.} $$

Summing moments about the platform revolute joint y_m axes and equating to zero gives

$$ {}^{m}M_{P_m(2)} = 0. \tag{6.1} $$

Thus there will be three scalar equations in terms of the components of the unknown reactions N_1, N_2 and N_3.

6.3.2 Summing the forces

The forces can be summed in the u_2, v_2 and w_2 directions and equated to zero to get three more independent scalar equations as follows

$$ \begin{aligned} \sum \text{Forces } \boldsymbol{u}_2 = {}&{}^{u_2}N_{1(1)} + {}^{u_2}N_{2(1)} + {}^{u_2}N_{3(1)} \\ &+ {}^{u_2}m_4 g_{(1)} + {}^{u_2}m_5 g_{(1)} + {}^{u_2}m_6 g_{(1)} \\ &+ {}^{u_2}m_p g_{(1)} + {}^{u_2}F_{p\ (1)} = 0, \end{aligned} \tag{6.2} $$

$$ \begin{aligned} \sum \text{Forces } \boldsymbol{v}_2 = {}&{}^{u_2}N_{1(2)} + {}^{u_2}N_{2(2)} + {}^{u_2}N_{3(2)} \\ &+ {}^{u_2}m_4 g_{(2)} + {}^{u_2}m_5 g_{(2)} + {}^{u_2}m_6 g_{(2)} \\ &+ {}^{u_2}m_p g_{(2)} + {}^{u_2}F_{p\ (2)} = 0, \end{aligned} \tag{6.3} $$

$$ \begin{aligned} \sum \text{Forces } \boldsymbol{w}_2 = {}&{}^{u_2}N_{1(3)} + {}^{u_2}N_{2(3)} + {}^{u_2}N_{3(3)} \\ &+ {}^{u_2}m_4 g_{(3)} + {}^{u_2}m_5 g_{(3)} + {}^{u_2}m_6 g_{(3)} \\ &+ {}^{u_2}m_p g_{(3)} + {}^{u_2}F_{p\ (3)} = 0 \end{aligned} \tag{6.4} $$

where

$$ {}^{u_2}\boldsymbol{N}_n = {}^{u_2}\mathbf{C}_m\ {}^{m}\boldsymbol{N}_n \qquad \text{n = 1,2,3 and m = n+3.} $$

6.3.3 Summing the moments

The moments can be summed about the u_2, v_2 and w_2 axes to formulate three further independent equations. The moments due to the weight of the upper arms and reaction forces about the u_2, v_2 and w_2 axes are given by

$$ {}^{u_2}\boldsymbol{M}_{O_p} = {}^{u_2}\overline{\boldsymbol{O_p p_{c_m}}} \times m_m\, {}^{u_2}\boldsymbol{g} + {}^{u_2}\overline{\boldsymbol{O_p B_n}} \times {}^{u_2}\boldsymbol{N}_n \qquad \text{n= 1,2,3 and m = n+3.} $$

Summing the moments about the u_2, v_2 and w_2 axes and equating to zero gives

$$ \sum \text{Moments } \boldsymbol{u}_2 = {}^{u_2}M_{O_p(1)} + {}^{u_2}M_{P(1)} = 0, \tag{6.5} $$

$$ \sum \text{Moments } \boldsymbol{v}_2 = {}^{u_2}M_{O_p(2)} + {}^{u_2}M_{P(2)} = 0, \tag{6.6} $$

$$ \sum \text{Moments } \boldsymbol{w}_2 = {}^{u_2}M_{O_p(3)} + {}^{u_2}M_{P(3)} = 0. \tag{6.7} $$

6.4 Solving for the reactions N

A symbolic manipulation package such as mathematica can be used to manipulate the nine scalar equations 6.1-6.7 so that they can be explicitly written in terms of the nine spherical joint reaction force components. These nine independent equations can then be written as an invertible 9×9 matrix and a 9×1 vector containing the nine spherical joint reaction force components to be obtained.

6.5 Calculation of the actuator torques

The next step is to calculate the moments about the **y** axes of the base revolute joints, i.e. the torques required from the motors to hold the mechanism in a particular position. Consider the forces on the lower arm 1; the forces of interest are shown below in figure 6.3.

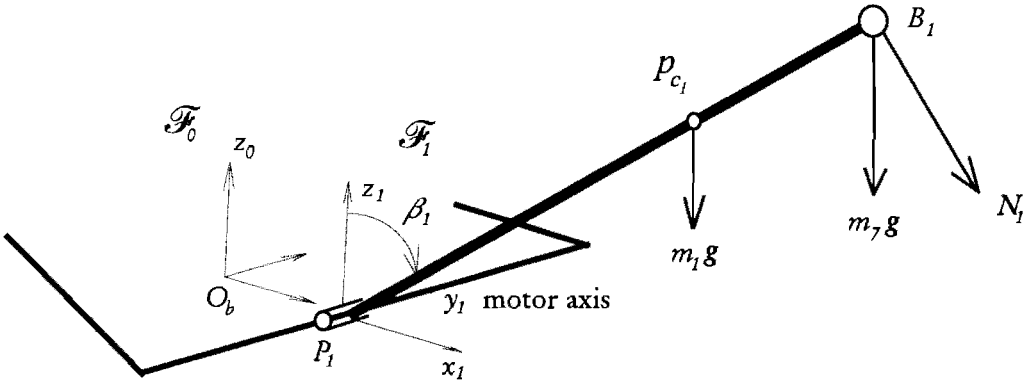


Figure 6.3 Forces acting on the lower arm 1.

To find the required motor torques to maintain a particular position moments are taken about the motor axes y_1, y_2 and y_3. The moments due to the weight of the arms, weight of the spherical joints and reaction forces are given by

$$ {}^{n}\boldsymbol{M}_{B_n} = {}^{n}\overline{\boldsymbol{P}_n\boldsymbol{p}_{c_n}} \times m_n {}^{n}\boldsymbol{g} + {}^{n}\overline{\boldsymbol{P}_n\boldsymbol{B}_n} \times m_i {}^{n}\boldsymbol{g} + {}^{n}\overline{\boldsymbol{P}_n\boldsymbol{B}_n} \times {}^{n}\boldsymbol{N}_n \qquad \text{n = 1,2,3 and i = n+6.} $$

Summing moments about the axes of the base revolute joints the motor torques are given by

$$ \boldsymbol{\tau}_n = {}^{n}M_{B_n(2)}. $$

6.6 Comparison with the experimental results

The static motor torques were measured experimentally by fixing two arms into a known set position and moving the third arm through a range of angles at intervals of 20°. At each interval a force spring balance was used to measure the force required to maintain the arm in that position. Thus the torque required to hold the arm in each position could be calculated. The mathematically modelled results are compared with the experimental results in figure 6.4.

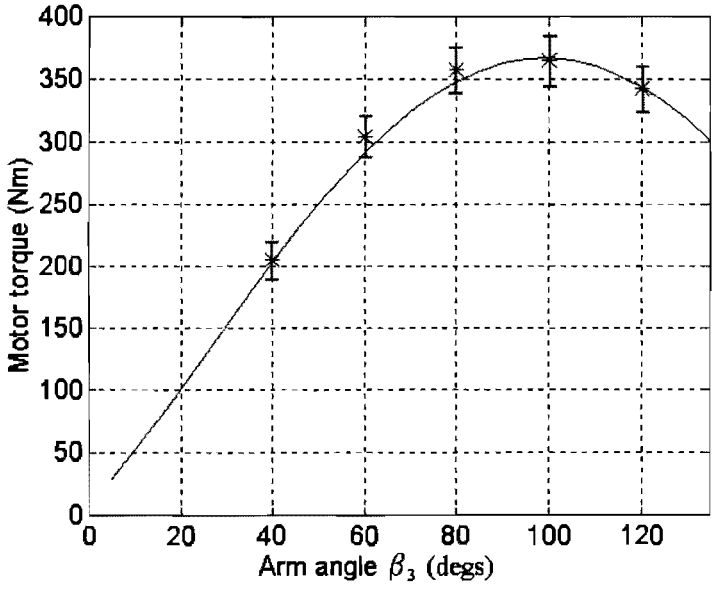


Figure 6.4 Comparison between modelled and experimental actuator torques.

In the case shown in figure 6.4 the arms one and two were set to 20° and 0° respectively while arm three was varied from 0° to 120° in steps of 20°. The experimental results compare favourably with the model, especially when taking into account the errors associated with the measurements, such as, maintaining a 90° angle between the spring balance and the arm ≈±3°, the accuracy of the balance ≈±3kg, the accuracy of the measured arm angle ≈±3° and the distance from the spring balance to the revolute joint axes ≈±20mm. These errors equate to an error in the experimentally measured torque ranging between 30 and 40 Nm depending on the magnitude of the torque. The mathematical model also assumes no friction effects between the joints.

6.7 Static model of the two dof mechanism

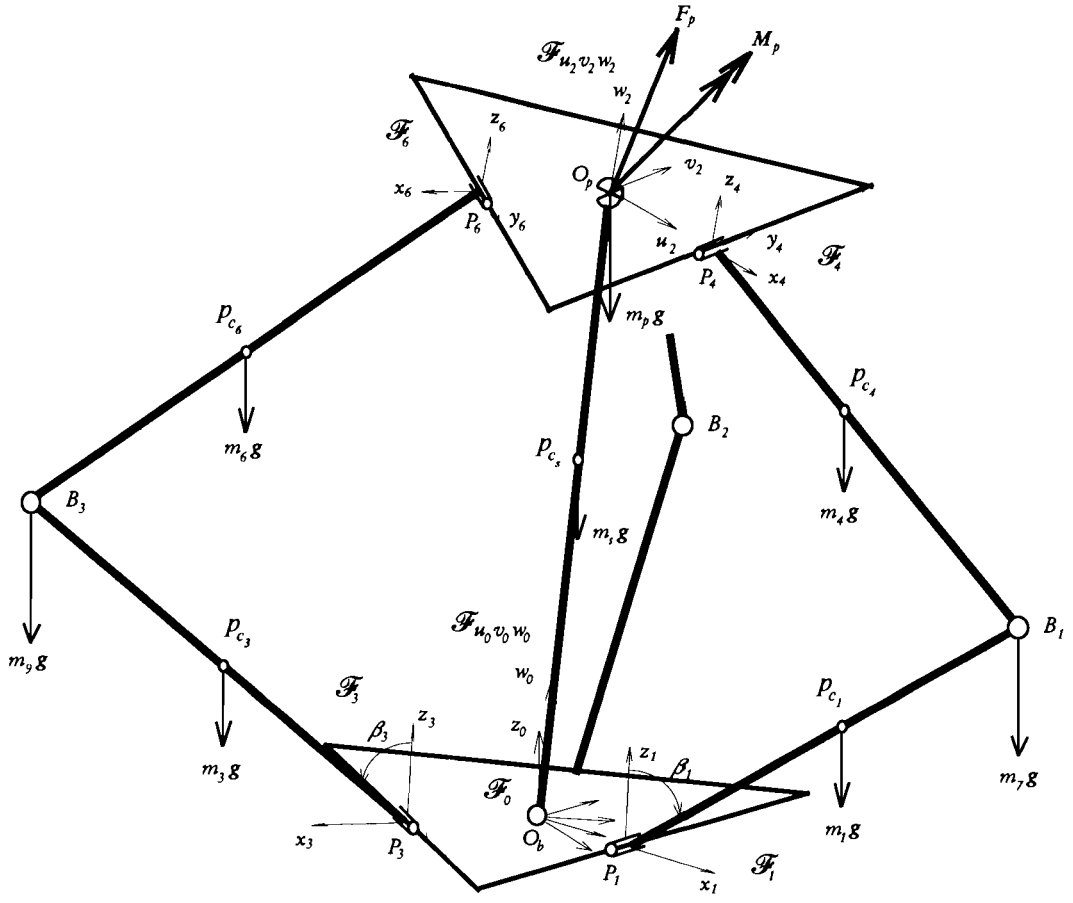


Figure 6.5 Model showing external forces acting on the two dof mechanism.

For the following static analysis the same assumptions of symmetry are made as for the model of the three dof mechanism. The arms and joints are modelled as for the three dof mechanism with the addition of a central strut having a mass centre at point p_{c_s}.

6.8 Formulation of the static equations of equilibrium

Given that the three lower arms are fixed in position, if an imaginary cut is made at the four spherical joints B_1, B_2, B_3 and O_b, the upper portion of the mechanism will form a statically determinant structure supported at the four spherical joints. The four spherical joint force reaction vectors N_1, N_2, N_3 and N_s can be split into three independent orthogonal components; therefore there will be 12 unknown reaction components.

Similar to the analysis for the three dof mechanism, three scalar equations can be formulated in terms of the reactions N_1, N_2 and N_3 due to the fact that moments cannot be sustained about the platform revolute joint y_m axes (m = 4,5,6). Six more independent equations can be formulated from the 6 equations of static equilibrium, i.e. summing the forces and moments and equating to zero, giving a total of 9 independent equations in terms of the unknown spherical joint reaction vector components.

Two more independent equations can be formulated in terms of the strut reactions N_s at O_b owing to the fact that moments cannot be sustained about the Hookes joint attached to the platform. The last equation can be formulated in terms of the reaction N_3 owing to the fact that moments cannot be sustained about the base revolute joint y_3 axis since the joint is not driven.

6.8.1 Summing the moments about the platform revolute joints

The moments can be summed about the platform revolute joints and equated to zero as in section 6.3.1. to give three independent equations in terms of the three spherical reactions N_1, N_2 and N_3.

6.8.2 Summing the forces

Three more equations can be formulated by summing the forces in the u_2, v_2 and w_2 directions as in sec 6.3.2.

$$\begin{aligned}\sum \text{Forces } \boldsymbol{u}_2 = {}^{u_2}N_{1(1)} + {}^{u_2}N_{2(1)} + {}^{u_2}N_{3(1)} + {}^{u_2}N_{s(1)} \\ + {}^{u_2}m_4 g_{(1)} + {}^{u_2}m_5 g_{(1)} + {}^{u_2}m_6 g_{(1)} \\ + {}^{u_2}m_p g_{(1)} + {}^{u_2}m_s g_{(1)} + {}^{u_2}F_{p\,(1)} = 0,\end{aligned}$$

$$\begin{aligned}\sum \text{Forces } \boldsymbol{v}_2 = {}^{u_2}N_{1(2)} + {}^{u_2}N_{2(2)} + {}^{u_2}N_{3(2)} + {}^{u_2}N_{s(2)} \\ + {}^{u_2}m_4 g_{(2)} + {}^{u_2}m_5 g_{(2)} + {}^{u_2}m_6 g_{(2)} \\ + {}^{u_2}m_p g_{(2)} + {}^{u_2}m_s g_{(2)}\,{}^{u_2}F_{p\,(2)} = 0,\end{aligned}$$

$$\begin{aligned}\sum \text{Forces } \boldsymbol{w}_2 = {}^{u_2}N_{1(3)} + {}^{u_2}N_{2(3)} + {}^{u_2}N_{3(3)} + {}^{u_2}N_{s(3)} \\ + {}^{u_2}m_4 g_{(3)} + {}^{u_2}m_5 g_{(3)} + {}^{u_2}m_6 g_{(3)} \\ + {}^{u_2}m_p g_{(3)} + {}^{u_2}m_s g_{(3)}\,{}^{u_2}F_{p\,(3)} = 0.\end{aligned}$$

6.8.3 Summing the moments

The moments can be summed about the u_2, v_2 and w_2 axes to formulate three further independent equations. The moments due to the weight of the upper arms, weight of the strut and the four reaction forces about the u_2, v_2 and w_2 axes are given by

$${}^{u_2}\boldsymbol{M}_{O_p} = {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{p}_{c_m}} \times m_m\,{}^{u_2}\boldsymbol{g} + {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{p}_{c_s}} \times m_s\,{}^{u_2}\boldsymbol{g} + {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{B}_n} \times {}^{u_2}\boldsymbol{N}_n + {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{O}_b} \times {}^{u_2}\boldsymbol{N}_s$$

n = 1,2,3 and m = n+3.

Summing the moments about the u_2, v_2 and w_2 axes and equating to zero gives

$$\sum \text{Moments } \boldsymbol{u}_2 = {}^{u_2}M_{O_p(1)} + {}^{u_2}M_{p(1)} = 0,$$

$$\sum \text{Moments } v_2 = {}^{u_2}M_{O_p}(2) + {}^{u_2}M_P(2) = 0,$$

$$\sum \text{Moments } w_2 = {}^{u_2}M_{O_p}(3) + {}^{u_2}M_P(3) = 0.$$

6.8.4 Summing the moments about the u_2 and v_2 axes

Two further independent equations can be formulated in terms of the strut reactions N_s at O_b owing to the fact that moments cannot be sustained about the Hookes joint attached to the platform. The moments about the u_2 and v_2 axes due to the strut reaction are given by

$${}^{u_2}\boldsymbol{M}_{O_p strut} = {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{p}_{c_s}} \times m_s {}^{u_2}\boldsymbol{g} + {}^{u_2}\overline{\boldsymbol{O}_p\boldsymbol{O}_b} \times {}^{u_2}\boldsymbol{N}_s.$$

Summing the moments about the u_2 and v_2 axes and equating to zero gives

$$\sum \text{Moments } \boldsymbol{u}_2 = {}^{u_2}M_{O_p strut}(1) = 0,$$

$$\sum \text{Moments } \boldsymbol{v}_2 = {}^{u_2}M_{O_p strut}(2) = 0.$$

6.8.5 Summing the moments about the base revolute joint

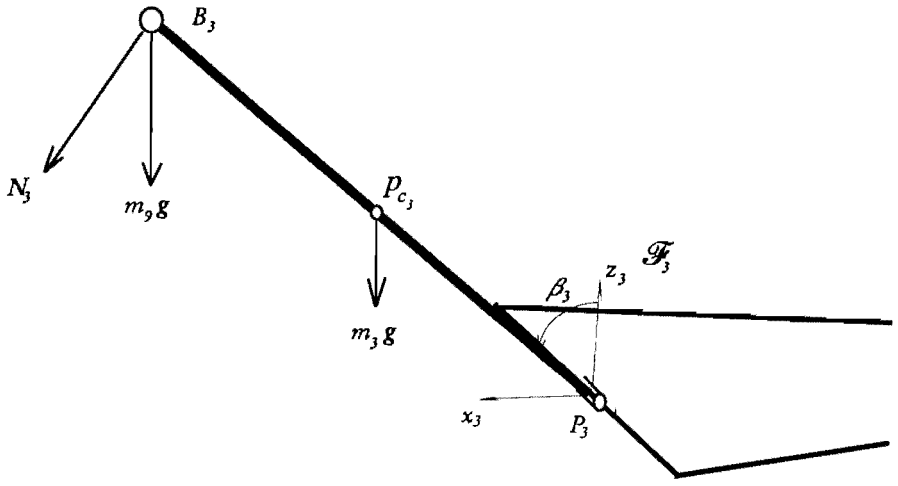


Figure 6.6 Forces acting on lower passive arm.

The last equation can be formulated in terms of the reaction N_3 by summing the moments about the bass revolute joint y_3 axis and equating to zero. Consider the forces acting on the lower arm three, the forces of interest are shown below in figure 6.6. The two forces of interest are the reaction vector N_3 and the gravity force $m_3 g$ due to the mass of the lower arm. The moment due to the weight of the arm and reaction force is given by

$${}^3\boldsymbol{M}_{P_3} = {}^3\overline{\boldsymbol{P}_3\boldsymbol{p}_{c_3}} \times m_3 {}^3\boldsymbol{g} + {}^3\overline{\boldsymbol{P}_3\boldsymbol{B}_3} \times {}^3\boldsymbol{N}_3$$

where

$${}^3\boldsymbol{N}_3 = {}^3\mathbf{C}_6 \, (-{}^6\boldsymbol{N}_3).$$

Equating the moment about the base revolute joint y_3 axis to zero gives

$${}^3M_{P_3}(2) = 0.$$

6.9 Solving for the reactions and calculation of the actuator torques

The reaction forces and actuator torques can be calculated as explained in sections 6.4 and 6.5. In this case there are twelve equations rather than nine.

6.10 Comparison with the experimental results

The static motor torques were measured experimentally by fixing one arm into a known set position and moving another arm through a range of angles at intervals of 20°. At each interval a force spring balance was used to measure the force required to maintain the arm in that position. Thus the torque required to hold the arm in each position could be calculated. The mathematically modelled results are compared with the experimental results in figure 6.7.

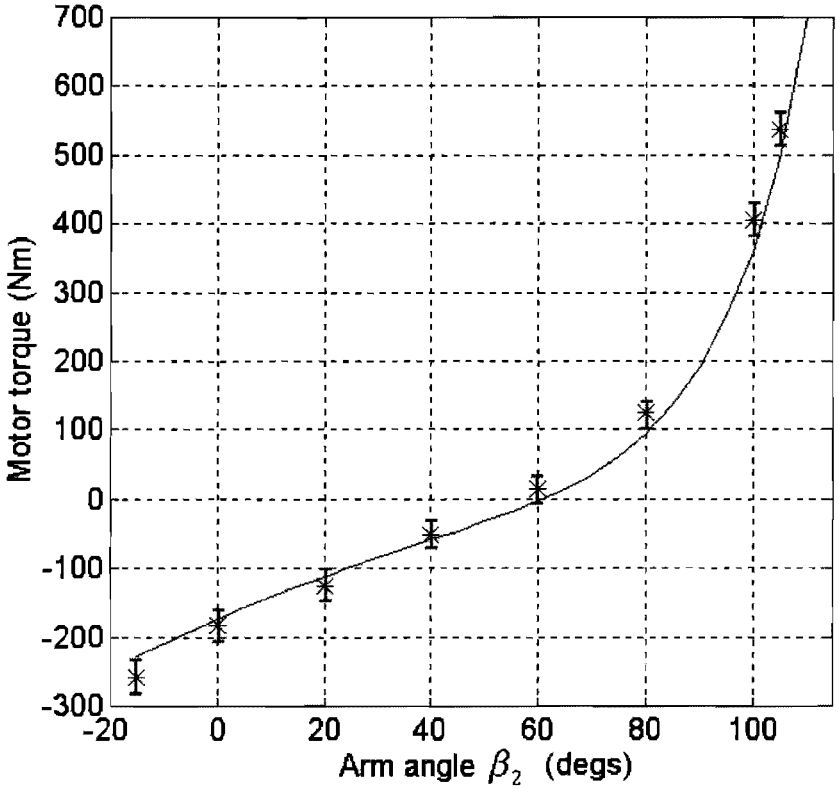


Figure 6.7 Comparison between modelled and experimental actuator torques.

In the case shown in figure 6.7 arm one was set to 20° while arm two was varied from -15° to 110°. The experimental results compare favourably with the model, especially when taking into account the errors associated with the experimental measurements. These errors equate to an error in the experimentally measured torque ranging between 30 and 45 Nm depending on the magnitude of the torque.

Chapter 7

Kinematics of an orbiting object

7.1 Formulation of the kinematics of an orbiting object for an inertial observer

In order that the dynamics of the tracking mechanism itself can be studied, an idealised tracking path is first modelled. Although this path is quite different from that of a real satellite it is used to demonstrate tracking movements such as a 0° elevation sweep about the horizon. It is assumed that the tracked object traces a perfect circular path and is constrained to lie in a plane perpendicular to the vector $\overline{O_I O_s}$ (c.f. figure 7.1).

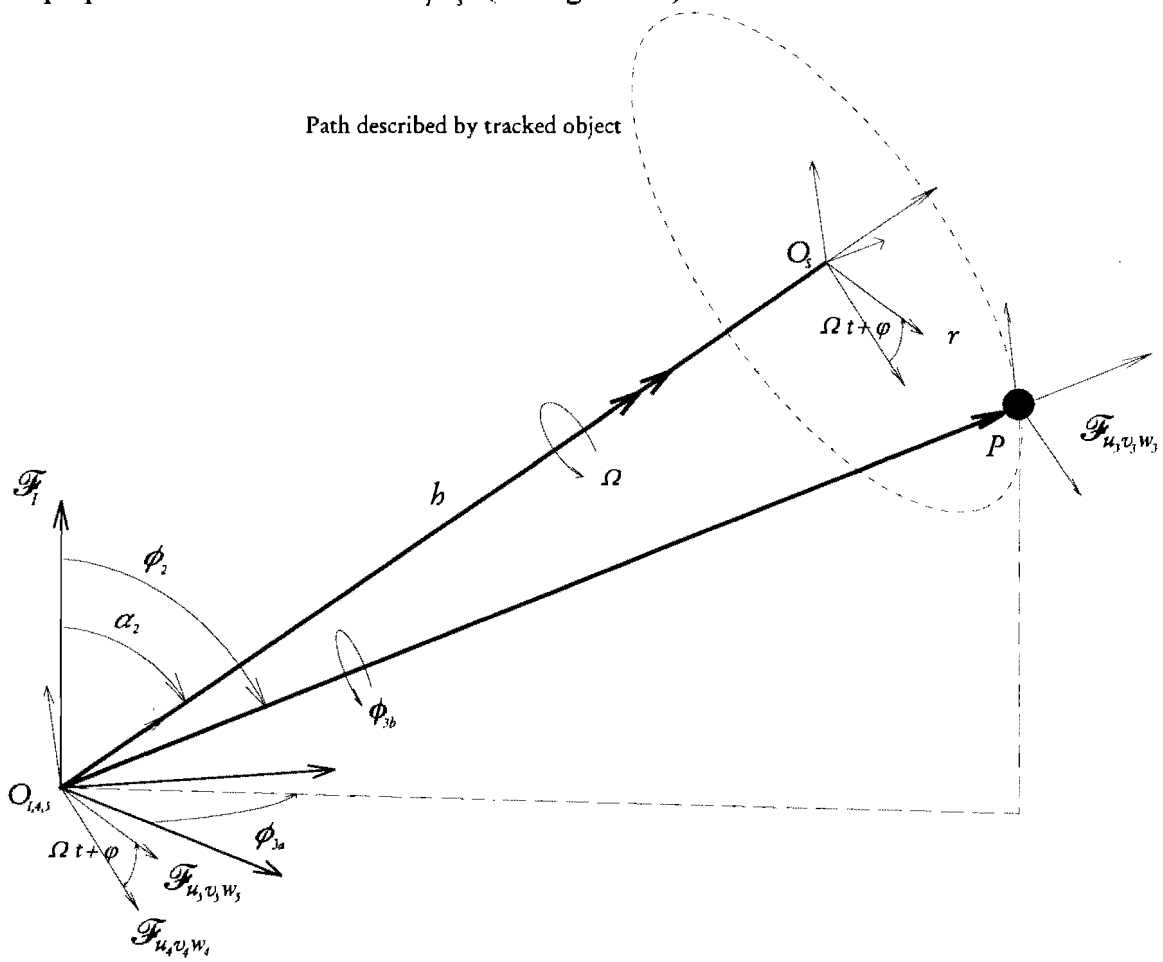


Figure 7.1 Coordinate systems used to describe an idealised satellite path.

The aiming coordinates of the tracked object P with respect to the inertial frame $\mathcal{F}_I$ are given by the traditional spherical coordinate angles denoted by ϕ_{3a} (a rotation about the 3 axis), ϕ_2 (a rotation about the moved 2 axis) and ϕ_{3b} (a rotation about the twice moved 3 axis, i.e. a

rotation about the line of sight). Although it is not strictly necessary to define the last angle ϕ_{3b} in order to completely specify the aiming coordinates of the tracked object, it is used to formulate the Euler rate projection matrices which are developed in section 7.1.2. The 3 axis of each of the two frames $\mathcal{F}_{u_4v_4w_4}$ and $\mathcal{F}_{u_5v_5w_5}$ (i.e. the axes w_4 and w_5) are aligned with the direction vector $\overline{O_I O_s}$. The frame $\mathcal{F}_{u_5v_5w_5}$ rotates with an angular speed of Ω about the 3 axis relative to frame $\mathcal{F}_{u_4v_4w_4}$. A body fixed frame $\mathcal{F}_{u_3v_3w_3}$ (fixed to the tracked object) has its 3 axis aligned with the direction $\overline{O_I P}$.

The position of the tracked object P expressed in frame $\mathcal{F}_{u_5v_5w_5}$ is given by

$$^{u_5}P = \begin{pmatrix} r \\ 0 \\ h \end{pmatrix}.$$

The relative position of the frames $\mathcal{F}_{u_4v_4w_4}$ and $\mathcal{F}_{u_5v_5w_5}$ are related by the rotation matrix $^{u_5}\mathbf{C}_{u_4}$

$$^{u_5}\mathbf{C}_{u_4} = \begin{bmatrix} c(\Omega t+\varphi) & s(\Omega t+\varphi) & 0 \\ -s(\Omega t+\varphi) & c(\Omega t+\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where φ is the initial angular offset at time $t = 0$ secs.

The relative position of the frames $\mathcal{F}_I$ and $\mathcal{F}_{u_4v_4w_4}$ are related by the rotation matrix $^{u_4}\mathbf{C}_I$

$$^{u_4}\mathbf{C}_I = \mathbf{C}_2(\alpha_2)\mathbf{C}_3(\alpha_3).$$

The position of the tracked object P can be expressed in the inertial frame $\mathcal{F}_I$ as

$$P = {}^I\mathbf{C}_{u_5}\,{}^{u_5}P \qquad \text{where } {}^I\mathbf{C}_{u_5} = {}^{u_5}\mathbf{C}_I^T \text{ and } {}^{u_5}\mathbf{C}_I = {}^{u_5}\mathbf{C}_{u_4}\,{}^{u_4}\mathbf{C}_I.$$

Note the vectors expressed in the inertial frame will not be preceded by a superscript ^{I}.

7.1.1 Calculation of the aiming angles ϕ with respect to frame $\mathcal{F}_I$

The angle ϕ_{3a} is given by

$$\phi_{3a} = \text{atan2}(P_{(2)}, P_{(1)}). \tag{7.1}$$

The angle ϕ_2 is given by

$$\phi_2 = \text{atan2}\left(\sqrt{P_{(1)}^2 + P_{(2)}^2}, P_{(3)}\right). \tag{7.2}$$

The aiming angle ϕ_{3b} can be found by noting that

$$\mathbf{C}_2(\eta)\mathbf{C}_3(\Omega t+\varphi)\mathbf{C}_2(\alpha_2)\mathbf{C}_3(\alpha_3) = \mathbf{C}_3(\phi_{3b})\mathbf{C}_2(\phi_2)\mathbf{C}_3(\phi_{3a}) \tag{7.3}$$

where η is the rotation about the 2 axis of the frame $\mathcal{F}_{u_5v_5w_5}$ needed to align frame $\mathcal{F}_{u_5v_5w_5}$ with frame $\mathcal{F}_{u_3v_3w_3}$. Equation (7.3) can be rearranged to get

$$\mathbf{C}_3(\phi_{3b}) = \mathbf{C}_2(\eta)\mathbf{C}_3(\Omega t+\varphi)\mathbf{C}_2(\alpha_2)\mathbf{C}_3(\alpha_3)\mathbf{C}_2^T(\phi_2)\mathbf{C}_3^T(\phi_{3a}).$$

The above can be written more fully as

$$\begin{bmatrix} c\phi_{3b} & s\phi_{3b} & 0 \\ -s\phi_{3b} & c\phi_{3b} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\eta & 0 & -s\eta \\ 0 & 1 & 0 \\ s\eta & 0 & c\eta \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

$$= \begin{bmatrix} C_{11}c\eta - C_{31}s\eta & C_{12}c\eta - C_{32}s\eta & C_{13}c\eta - C_{33}s\eta \\ C_{21} & C_{22} & C_{23} \\ C_{11}s\eta + C_{31}c\eta & C_{12}s\eta + C_{32}c\eta & C_{13}s\eta + C_{33}c\eta \end{bmatrix}.$$

The aiming angle ϕ_{3b} is given by

$$\phi_{3b} = \text{atan2}(-C_{21}, C_{22}).$$

Note: from above it can be seen that ϕ_{3b} is a function of the angles $\Omega, \varphi, \alpha_2, \alpha_3, \phi_{3a}$ and ϕ_2. Although ϕ_{3b} is not strictly needed to specify the pointing direction of the satellite it is needed to formulate the Euler rate projection matrix ${}^{u_3}\mathbf{S}_I(\phi)$ mapping the angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ into the Euler rates.

7.1.2 Calculation of the rates and time derivatives $\dot{\phi}, \ddot{\phi}$

It is now possible, although tedious, to find the time derivatives of ϕ_2 and ϕ_{3a} by differentiating equations (7.1) and (7.2) with respect to time. An alternative method is to utilise the inverse of the Euler rate projection matrix $\mathbf{S}^{-1}$ to project the angular velocity $\underline{\omega}_{u_3,I}$ of the tracked object (angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$) relative to the inertial frame $\mathcal{F}_I$ into the Euler rates.

The angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ relative to the inertial frame $\mathcal{F}_I$ can be written as

$$\begin{aligned} \underline{\omega}_{u_3,I} &= \underline{\omega}_{u_3,u_5} + \underline{\omega}_{u_5,u_4} + \underline{\omega}_{u_4,I} \\ &= \underline{\omega}_{u_5,u_4} \qquad \text{since} \quad \underline{\omega}_{u_3,u_5} = \underline{\omega}_{u_4,I} = 0 \end{aligned}$$

or in component form as

$${}^{u_3}\omega_{u_3,I} = {}^{u_3}\mathbf{C}_{u_5}\, {}^{u_5}\omega_{u_3,u_4} \tag{7.4}$$

where

$${}^{u_3}\omega_{u_3,I} = {}^{u_3}\mathbf{S}_I\dot{\phi}$$

and the angular velocity of frame $\mathcal{F}_{u_5v_5w_5}$ with respect to frame $\mathcal{F}_{u_4v_4w_4}$ is given by

$${}^{u_5}\omega_{u_5,u_4} = \begin{pmatrix} 0 \\ 0 \\ \Omega \end{pmatrix}.$$

The rotation matrix ${}^{u_3}\mathbf{C}_{u_5}$ can be found as follows:

$${}^{u_3}\mathbf{C}_{u_5} = {}^{u_3}\mathbf{C}_I\, {}^{u_4}\mathbf{C}_I^T\, {}^{u_5}\mathbf{C}_{u_4}^T$$

where

$${}^{u_3}\mathbf{C}_I = \mathbf{C}_3(\phi_{3b})\mathbf{C}_2(\phi_2)\mathbf{C}_3(\phi_{3a}).$$

The 3-2-3 Euler rate projection matrix ${}^{u_3}\mathbf{S}_I$ is given by

$$ {}^{u_3}\mathbf{S}_I = \begin{bmatrix} 0 & s\phi_{3b} & -c\phi_{3b}s\phi_2 \\ 0 & c\phi_{3b} & s\phi_2 s\phi_{3b} \\ 1 & 0 & c\phi_2 \end{bmatrix}. $$

Equation (7.4) can now be written as

$$ {}^{u_3}\mathbf{S}_I \dot{\phi} = {}^{u_3}\mathbf{C}_{u_5}\, {}^{u_5}\omega_{u_5,u_4}. \tag{7.5} $$

The Euler rates are then given by

$$ \dot{\phi} = {}^{u_3}\mathbf{S}_I^{-1}\, {}^{u_3}\mathbf{C}_{u_5}\, {}^{u_5}\omega_{u_5,u_4} $$

where

$$ {}^{u_3}\mathbf{S}_I^{-1} = \begin{bmatrix} \dfrac{c\phi_{3b}c\phi_2}{s\phi_2} & \dfrac{-s\phi_{3b}c\phi_2}{s\phi_2} & 1 \\ s\phi_{3b} & c\phi_{3b} & 0 \\ \dfrac{-c\phi_{3b}}{s\phi_2} & \dfrac{s\phi_{3b}}{s\phi_2} & 0 \end{bmatrix}. $$

Differentiating (7.5) with respect to time, the time derivatives of the pointing rates are given by

$$ {}^{u_3}\dot{\mathbf{S}}_I \dot{\phi} + {}^{u_3}\mathbf{S}_I \ddot{\phi} = {}^{u_3}\dot{\mathbf{C}}_{u_5}\, {}^{u_5}\omega_{u_5,u_4} + {}^{u_3}\mathbf{C}_{u_5}\, {}^{u_5}\dot{\omega}_{u_5,u_4}. $$

Noting that ${}^{u_3}\dot{\mathbf{C}}_{u_5} = \mathbf{O}$ and ${}^{u_3}\dot{\omega}_{u_3,u_4} = 0$ the time derivative of the Euler rates is then given by

$$ \ddot{\phi} = {}^{u_3}\mathbf{S}_I^{-1}(-{}^{u_3}\dot{\mathbf{S}}_I \dot{\phi}) $$

where the time derivative of the 3-2-3 Euler rate projection matrix is given by

$$ {}^{u_3}\dot{\mathbf{S}}_I = \begin{bmatrix} 0 & \dot{\phi}_{3b}c\phi_{3b} & \dot{\phi}_{3b}s\phi_{3b}s\phi_2 - \dot{\phi}_2 c\phi_{3b}c\phi_2 \\ 0 & -\dot{\phi}_{3b}s\phi_{3b} & \dot{\phi}_2 c\phi_2 s\phi_{3b} + \dot{\phi}_{3b}s\phi_2 c\phi_{3b} \\ 0 & 0 & -\dot{\phi}_2 s\phi_2 \end{bmatrix}. $$

7.2 Multi degree of freedom base motions

It was envisaged that the tracking mechanism might be mounted on a moving ship. Since the mechanism's working range is restricted to the upper hemisphere, provision is made for a stabilised platform. Note that a stabilised platform would not be required if a reduced tracking coverage (e.g. from 10° above the horizon) is acceptable in heavy weather.

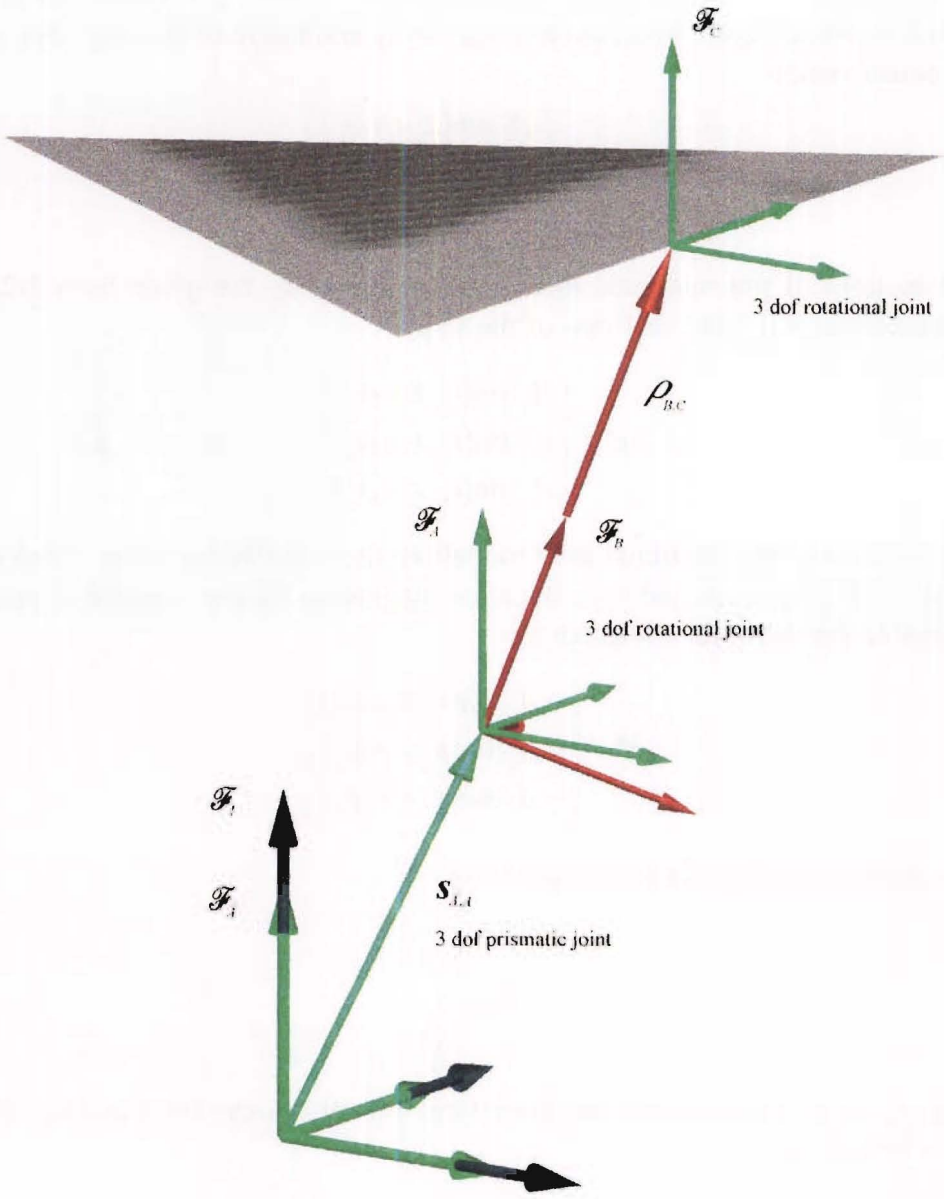


Figure 7.2 Coordinate frames describing the ship and stabilised platform.

The ship and stabilised platform were modelled by three multi dof joints (c.f. figure 7.2), the first being a three dof prismatic joint sited at frame $\mathcal{F}_{\hat{A}}$, the second being a three dof rotational joint sited at frame $\mathcal{F}_{A}$, and the third being another three dof rotational joint used to model the stabilised platform sited at frame $\mathcal{F}_{C}$. The relative motion between these two frames is described by sinusoidal functions. Although this model may not exactly replicate the motion of a ship it allows for a fairly close approximation.

The translatory motions of the prismatic joint sited at frame $\mathcal{F}_{\hat{A}}$ are along the principal axes of the inertial frame and correspond to the surge, sway and heave of the ship; they are given by the displacement vector

$$ {}^{A}\boldsymbol{s}_{\hat{A}A} = \begin{pmatrix} A_{t_1} \sin(\varepsilon_{t_1} t + \phi_{t_1}) \\ A_{t_2} \sin(\varepsilon_{t_2} t + \phi_{t_2}) \\ A_{t_3} \sin(\varepsilon_{t_3} t + \phi_{t_3}) \end{pmatrix}. \tag{7.6} $$

The rotary motions of the rotational joint sited at frame $\mathcal{F}_{A}$ are given by a 3-2-1 Euler set corresponding to the roll, pitch and yaw of the ship.

$$ \boldsymbol{\psi} = \begin{pmatrix} A_{r_1} \sin(\varepsilon_{r_1} t + \phi_{r_1}) \\ A_{r_2} \sin(\varepsilon_{r_2} t + \phi_{r_2}) \\ A_{r_3} \sin(\varepsilon_{r_3} t + \phi_{r_3}) \end{pmatrix}. \tag{7.7} $$

The rotary motions of the rotational joint modelling the stabilised platform sited at frame $\mathcal{F}_{C}$ are given by a 1-2-3 Euler set and have the same magnitude but are opposite in sign to the roll, pitch and yaw of the ship; they are given by

$$ \boldsymbol{\vartheta} = \begin{pmatrix} -A_{r_3} \sin(\varepsilon_{r_3} t + \phi_{r_3}) \\ -A_{r_2} \sin(\varepsilon_{r_2} t + \phi_{r_2}) \\ -A_{r_1} \sin(\varepsilon_{r_1} t + \phi_{r_1}) \end{pmatrix}. \tag{7.8} $$

Should a stabilised platform not be included then

$$ \boldsymbol{\vartheta} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{7.9} $$

The rates pertaining to these joints and their time derivatives can be found by differentiating with respect to time.

7.3 Calculation of the aiming angles φ with respect to frame $\mathcal{F}_C$

If a stabilised platform is not included, a new set of aiming angles $\varphi_{3b}, \varphi_2, \varphi_{3a}$ needs to be calculated with respect to the translating and rotating frame $\mathcal{F}_C$, as it is from this frame that the robot must track the satellite. Referring to figures 7.1 and 7.2 it can be seen that the relationship between the frames is given by

$$^{u_3}\mathbf{C}_C(\varphi) = {}^{u_3}\mathbf{C}_I(\phi)\ {}^{A}\mathbf{C}_I^T\ {}^{B}\mathbf{C}_A^T(\psi)\ {}^{C}\mathbf{C}_B^T(\vartheta) \qquad {}^{A}\mathbf{C}_I^T = \mathbf{1}. \tag{7.10}$$

For a stabilised platform equation (7.10) becomes

$$^{u_3}\mathbf{C}_C(\varphi) = {}^{u_3}\mathbf{C}_I(\phi).$$

Thus the aiming angles φ are given by

$$\varphi_{3b} = \phi_{3b},\ \varphi_2 = \phi_2,\ \varphi_{3a} = \phi_{3a}.$$

Without the stabilised platform equation (7.10) becomes

$$^{u_3}\mathbf{C}_C(\varphi) = {}^{u_3}\mathbf{C}_I(\phi)\ {}^{B}\mathbf{C}_A^T(\psi). \tag{7.11}$$

Expanding (7.11) gives

$$\begin{bmatrix} c\varphi_{3b}c\varphi_2c\varphi_{3a} - s\varphi_{3b}s\varphi_{3a} & c\varphi_{3b}c\varphi_2s\varphi_{3a} + s\varphi_{3b}c\varphi_{3a} & -c\varphi_{3b}s\varphi_2 \\ -s\varphi_{3b}c\varphi_2c\varphi_{3a} - c\varphi_{3b}s\varphi_{3a} & -s\varphi_{3b}c\varphi_2s\varphi_{3a} + c\varphi_{3b}c\varphi_{3a} & s\varphi_{3b}s\varphi_2 \\ s\varphi_2c\varphi_{3a} & s\varphi_2s\varphi_{3a} & c\varphi_2 \end{bmatrix} = \mathbf{C}(\phi, \psi).$$

Equating coefficients the aiming angles φ can be found as a function of the known angles ϕ, ψ. The angle φ_2 is given by

$$\varphi_2 = \text{acos}(C_{33}) \qquad 0° \le \varphi_2 \le 180°. \tag{7.12}$$

The angle φ_{3a} is given by

$$\varphi_{3a} = \text{atan2}(C_{32}, C_{31}) \tag{7.13}$$

and the angle φ_{3b} is given by

$$\varphi_{3b} = \text{atan2}(C_{23}, -C_{13}). \tag{7.14}$$

7.3.1 Calculation of the rates and time derivatives $\dot{\varphi}$, $\ddot{\varphi}$

It is now possible, although tedious to find the time derivatives of $\varphi_{3b}, \varphi_2, \varphi_{3a}$ by differentiating equations (7.12), (7.13) and (7.14) with respect to time. As done before in section 7.1.2, the alternative method is to utilise the inverse of the Euler rate projection matrix $\mathbf{S}^{-1}$ to project the angular velocity $\underline{\omega}_{u_3,I}$ of the tracked object (angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$) relative to the inertial frame $\mathcal{F}_I$ in to the Euler rates.

The angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ relative to the inertial frame $\mathcal{F}_I$ can be written as

$$\begin{aligned}\underline{\omega}_{u_3,I} &= \underline{\omega}_{u_3,C} + \underline{\omega}_{C,B} + \underline{\omega}_{B,A} + \underline{\omega}_{A,I} \\ &= \underline{\omega}_{u_3,C} + \underline{\omega}_{C,B} + \underline{\omega}_{B,A} \qquad \text{since} \quad \underline{\omega}_{A,I} = 0\end{aligned}$$

or in component form as

$${}^{u_3}\omega_{u_3,I} = {}^{u_3}\omega_{u_3,C} + {}^{u_3}\mathbf{C}_C\,{}^{C}\omega_{C,B} + {}^{u_3}\mathbf{C}_B\,{}^{B}\omega_{B,A} \tag{7.15}$$

where the angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ with respect to frame $\mathcal{F}_I$ is given by

$${}^{u_3}\omega_{u_3,I} = {}^{u_3}\mathbf{S}_I\,\dot{\boldsymbol{\phi}}.$$

The angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ with respect to frame $\mathcal{F}_C$ is given by

$$\begin{aligned}{}^{u_3}\omega_{u_3,C} &= {}^{u_3}\mathbf{S}_C\,\dot{\boldsymbol{\varphi}} \\ &= \begin{bmatrix} 0 & s\varphi_{3b} & -c\varphi_{3b}s\varphi_2 \\ 0 & c\varphi_{3b} & s\varphi_2 s\varphi_{3b} \\ 1 & 0 & c\varphi_2 \end{bmatrix}\begin{pmatrix}\dot{\varphi}_{3b}\\ \dot{\varphi}_2\\ \dot{\varphi}_{3a}\end{pmatrix}\end{aligned}$$

and the angular velocity of frame $\mathcal{F}_C$ with respect to frame $\mathcal{F}_B$ is given by

$$\begin{aligned}{}^{C}\omega_{C,B} &= {}^{C}\mathbf{S}_B\,\dot{\boldsymbol{\vartheta}} \\ &= \begin{bmatrix} 0 & s\vartheta_{(1)} & c\vartheta_{(1)}c\vartheta_{(2)} \\ 0 & c\vartheta_{(1)} & -s\vartheta_{(1)}c\vartheta_{(2)} \\ 1 & 0 & s\vartheta_{(2)} \end{bmatrix}\begin{pmatrix}\dot{\vartheta}_{(1)}\\ \dot{\vartheta}_{(2)}\\ \dot{\vartheta}_{(3)}\end{pmatrix}\end{aligned}$$

and the angular velocity of frame $\mathcal{F}_B$ with respect to frame $\mathcal{F}_A$ is given by

$$\begin{aligned}{}^{B}\omega_{B,A} &= {}^{B}\mathbf{S}_A\,\dot{\boldsymbol{\psi}} \\ &= \begin{bmatrix} 1 & 0 & -s\psi_{(2)} \\ 0 & c\psi_{(1)} & s\psi_{(1)}c\psi_{(2)} \\ 0 & -s\psi_{(1)} & c\psi_{(1)}c\psi_{(2)} \end{bmatrix}\begin{pmatrix}\dot{\psi}_{(1)}\\ \dot{\psi}_{(2)}\\ \dot{\psi}_{(3)}\end{pmatrix}.\end{aligned}$$

Substituting for the above into equation (7.16) gives

$${}^{u_3}\mathbf{S}_I\,\dot{\boldsymbol{\phi}} = {}^{u_3}\mathbf{S}_C\,\dot{\boldsymbol{\varphi}} + {}^{u_3}\mathbf{C}_C\,{}^{C}\mathbf{S}_B\,\dot{\boldsymbol{\vartheta}} + {}^{u_3}\mathbf{C}_B\,{}^{B}\mathbf{S}_A\,\dot{\boldsymbol{\psi}}. \tag{7.16}$$

The Euler pointing rates for a stabilised platform are given by

$$\dot{\boldsymbol{\varphi}} = {}^{u_3}\mathbf{S}_C^{-1}\,{}^{u_3}\mathbf{S}_I\,\dot{\boldsymbol{\phi}}$$

and the Euler pointing rates for a non stabilised platform are given by

$$\dot{\boldsymbol{\varphi}} = {}^{u_3}\mathbf{S}_C^{-1}\left({}^{u_3}\mathbf{S}_I\,\dot{\boldsymbol{\phi}} - {}^{u_3}\mathbf{C}_B\,{}^{B}\mathbf{S}_A\,\dot{\boldsymbol{\psi}}\right)$$

where the inverse of the Euler projection matrix ${}^{u_3}\mathbf{S}_C^{-1}$ is given by

$$ {}^{u_3}\mathbf{S}_C^{-1} = \begin{bmatrix} \dfrac{c\varphi_{3b}c\varphi_2}{s\varphi_2} & \dfrac{-s\varphi_{3b}c\varphi_2}{s\varphi_2} & 1 \\ s\varphi_{3b} & c\varphi_{3b} & 0 \\ \dfrac{-c\varphi_{3b}}{s\varphi_2} & \dfrac{s\varphi_{3b}}{s\varphi_2} & 0 \end{bmatrix}. $$

To find the time derivatives of the rates differentiate equation (7.16) with respect to time get

$$ \begin{aligned} {}^{u_3}\dot{\mathbf{S}}_1\dot{\boldsymbol{\phi}} + {}^{u_3}\mathbf{S}_1\ddot{\boldsymbol{\phi}} &= {}^{u_3}\dot{\mathbf{S}}_C\dot{\boldsymbol{\varphi}} + {}^{u_3}\mathbf{S}_C\ddot{\boldsymbol{\varphi}} + {}^{u_3}\dot{\mathbf{C}}_C{}^{C}\mathbf{S}_B\dot{\boldsymbol{\vartheta}} + {}^{u_3}\mathbf{C}_C\left({}^{C}\dot{\mathbf{S}}_B\dot{\boldsymbol{\vartheta}} + {}^{C}\mathbf{S}_B\ddot{\boldsymbol{\vartheta}}\right) \\ &\quad + {}^{u_3}\dot{\mathbf{C}}_B{}^{B}\mathbf{S}_A\dot{\boldsymbol{\psi}} + {}^{u_3}\mathbf{C}_B\left({}^{B}\dot{\mathbf{S}}_A\dot{\boldsymbol{\psi}} + {}^{B}\mathbf{S}_A\ddot{\boldsymbol{\psi}}\right) \end{aligned} \tag{7.17} $$

where the time derivative of the rotation matrix ${}^{u_3}\dot{\mathbf{C}}_C$ is given by

$$ {}^{u_3}\dot{\mathbf{C}}_C = -{}^{u_3}\boldsymbol{\omega}^{\times}_{u_3,C}{}^{u_3}\mathbf{C}_C. $$

The time derivative of the rotation matrix ${}^{u_3}\dot{\mathbf{C}}_B$ is given by

$$ {}^{u_3}\dot{\mathbf{C}}_B = -{}^{u_3}\boldsymbol{\omega}^{\times}_{u_3,B}{}^{u_3}\mathbf{C}_B $$

where the angular velocity of frame $\mathcal{F}_{u_3v_3w_3}$ with respect to frame $\mathcal{F}_B$ is given by

$$ {}^{u_3}\boldsymbol{\omega}_{u_3,B} = {}^{u_3}\boldsymbol{\omega}_{u_3,C} + {}^{u_3}\mathbf{C}_C{}^{C}\boldsymbol{\omega}_{C,B}. $$

The time derivative of the Euler rate projection matrix ${}^{u_3}\mathbf{S}_C$ is given by

$$ {}^{u_3}\dot{\mathbf{S}}_C = \begin{bmatrix} 0 & \dot{\varphi}_{3b}c\varphi_{3b} & \dot{\varphi}_{3b}s\varphi_{3b}s\varphi_2 - \dot{\varphi}_2c\varphi_{3b}c\varphi_2 \\ 0 & -\dot{\varphi}_{3b}s\varphi_{3b} & \dot{\varphi}_2c\varphi_2s\varphi_{3b} + \dot{\varphi}_{3b}s\varphi_2c\varphi_{3b} \\ 0 & 0 & -\dot{\varphi}_2s\varphi_2 \end{bmatrix} $$

and time derivative of the Euler rate projection matrix ${}^{C}\mathbf{S}_B$ is given by

$$ {}^{C}\dot{\mathbf{S}}_B = \begin{bmatrix} 0 & \dot{\vartheta}_{(1)}c\vartheta_{(1)} & -\dot{\vartheta}_{(1)}s\vartheta_{(1)}c\vartheta_{(2)} - \dot{\vartheta}_{(2)}c\vartheta_{(1)}s\vartheta_{(2)} \\ 0 & -\dot{\vartheta}_{(1)}s\vartheta_{(1)} & -\dot{\vartheta}_{(1)}c\vartheta_{(1)}c\vartheta_{(2)} + \dot{\vartheta}_{(2)}s\vartheta_{(1)}s\vartheta_{(2)} \\ 0 & 0 & \dot{\vartheta}_{(2)}c\vartheta_{(2)} \end{bmatrix} $$

and time derivative of the Euler rate projection matrix ${}^{B}\mathbf{S}_A$ is given by

$$ {}^{B}\dot{\mathbf{S}}_A = \begin{bmatrix} 0 & 0 & -\dot{\psi}_{(2)}c\psi_{(2)} \\ 0 & -\dot{\psi}_{(1)}s\psi_{(1)} & \dot{\psi}_{(1)}c\psi_{(1)}c\psi_{(2)} - \dot{\psi}_2s\psi_{(1)}s\psi_{(2)} \\ 0 & -\dot{\psi}_{(1)}c\psi_{(1)} & -\dot{\psi}_{(1)}s\psi_{(1)}c\psi_{(2)} - \dot{\psi}_{(2)}c\psi_{(1)}s\psi_{(2)} \end{bmatrix}. $$

Rearranging (7.17) and substituting for the above $\ddot{\varphi}$ can be found for a stabilised platform

$$ \ddot{\boldsymbol{\varphi}} = {}^{u_3}\mathbf{S}_C^{-1}\left({}^{u_3}\dot{\mathbf{S}}_1\dot{\boldsymbol{\phi}} + {}^{u_3}\mathbf{S}_1\ddot{\boldsymbol{\phi}}\right) $$

and for a non stabilised platform

$$ \ddot{\boldsymbol{\varphi}} = {}^{u_3}\mathbf{S}_C^{-1}\left({}^{u_3}\dot{\mathbf{S}}_1\dot{\boldsymbol{\phi}} + {}^{u_3}\mathbf{S}_1\ddot{\boldsymbol{\phi}} - {}^{u_3}\dot{\mathbf{S}}_C\dot{\boldsymbol{\varphi}} - {}^{u_3}\dot{\mathbf{C}}_B{}^{B}\mathbf{S}_A\dot{\boldsymbol{\psi}} - {}^{u_3}\mathbf{C}_B\left({}^{B}\dot{\mathbf{S}}_A\dot{\boldsymbol{\psi}} + {}^{B}\mathbf{S}_A\ddot{\boldsymbol{\psi}}\right)\right). $$

7.4 The equivalence between the aiming angles φ of the tracked object and the pointing angles of the platform θ

In order to simplify the calculation of the pointing angles θ_{3a} and θ_2 of the platform, the assumption is made that these angles are equivalent to the aiming angles φ_{3a} and φ_2 of the tracked object w.r.t. the frame $\mathcal{F}_C$. This assumption is valid if it is assumed that the tracked object is at a large distance from the tracking mechanism: the direction vector $\overrightarrow{O_C P}$ (c.f. figure 7.3) is then parallel to the w_2 axis of the platform frame $\mathcal{F}_{u_2 v_2 w_2}$. Hence the aiming angles of the tracked object φ_{3a} and φ_2 w.r.t. the frame $\mathcal{F}_C$ are equal to the pointing angles θ_{3a} and θ_2 of the platform.

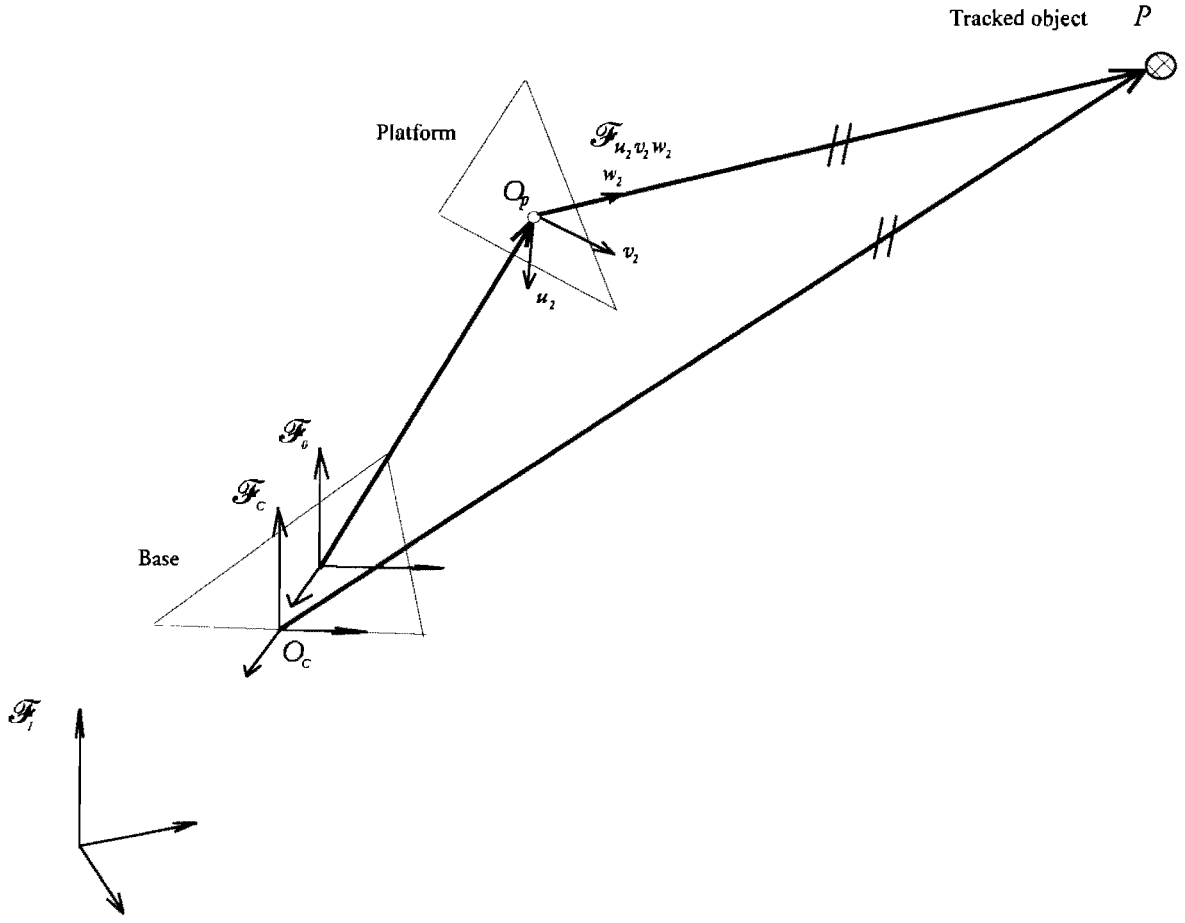


Figure 7.3 The equivalence between the pointing direction of the platform and the aiming direction of the tracked object w.r.t. the frame $\mathcal{F}_C$.

The position, velocity and acceleration kinematics of a orbiting object (with some constraints outlined in section 7.1) have been derived in this chapter. The pointing angles and rates of the tracked object can be used in conjunction with the inverse dynamics of the tracking mechanisms developed in Chapter 10 to simulate a tracking procedure.

Chapter 8

Dynamic analysis of open chains with rotational and/or prismatic joints

8.1 Introduction

The dynamic equations for open chains with rotational and/or prismatic joints are derived in this chapter. These equations will be used in Chapter 9 to solve the inverse dynamics problem. The approach used here largely follows the work of Hughes, Sincarsin and D'Eleuterio [1985,1988,1989]. Those familiar with this work could skip this chapter as the notation used here is fairly similar with only few departures. Only the equations required are developed here, and it is by no means a complete treatment of the problem. The references to Hughes *et al.* contain a more comprehensive course on the subject. Hughes and Sincarsin [1989] noted that although the basic equations of motion have been precisely known for hundreds of years and therefore there is no fundamental difficulty in writing out motion equations for a mechanism, problems arise due to choice of a convenient notation and organisation of the many subservient relationships into a cohesive whole. This chapter introduces a concise notation and organisation for describing the dynamics of multibody systems.

Given the motion of the mechanism the solution of the inverse dynamics leads to the divulgence of the actuated joint forces and/or the generalised forces acting between the links of the mechanism. Various formulations have been proposed based on principles variously attributed to Newton, Euler, Lagrange, Hamilton and D'Alembert. Although all of the formulations calculate the dynamics of a system, each has its advantages and disadvantages and there is no clear 'winner' for all problems. For example, the Langrangian approach has the advantage in that it is conducive for control implementation but it is unable to calculate the joint reactions (constraint forces). Not only do the approaches differ, but there are different formulations derived from the approaches, i.e. the recursive versus global approach. The computational efficiency of some of the different formulations is compared in Li and Sankar [1992,1993]. The formulation used here to solve the inverse dynamics is based upon a Newton-Euler approach. This particular approach was chosen since unlike the work/energy approaches, the constraint forces are calculated which is very important to a designer.

The analysis in this chapter is only applicable to open chains with rotational and/or prismatic joints. Many of the results obtained in this chapter are used in Chapter 9 where the dynamics of multiple closed looped mechanisms is presented.

8.2 Motion equations for a body

The motion equations for a body will not be derived here. Following the approach of Hughes, Sincarsin and D'Eleuterio [1988,1989] the motion equations for a body can be written as follows:

$$\begin{bmatrix} m_n\mathbf{1} & -{}^n\boldsymbol{c}_n^{\times} \\ {}^n\boldsymbol{c}_n^{\times} & {}^n\boldsymbol{J}_n \end{bmatrix}\begin{pmatrix} {}^n\dot{\mathbf{v}}_n \\ {}^n\dot{\omega}_n \end{pmatrix}+\begin{bmatrix} {}^n\omega_n^{\times} & \mathbf{O} \\ {}^n\mathbf{v}_n^{\times} & {}^n\omega_n^{\times} \end{bmatrix}\begin{bmatrix} m_n\mathbf{1} & -{}^n\boldsymbol{c}_n^{\times} \\ {}^n\boldsymbol{c}_n^{\times} & {}^n\boldsymbol{J}_n \end{bmatrix}\begin{pmatrix} {}^n\mathbf{v}_n \\ {}^n\omega_n \end{pmatrix}=\begin{pmatrix} {}^n\boldsymbol{F}_{n,T} \\ {}^n\boldsymbol{G}_{n,T} \end{pmatrix} \tag{8.1}$$

or

$${}^n\boldsymbol{M}_n\,{}^n\dot{\boldsymbol{v}}_n + {}^n\boldsymbol{v}_n^{\otimes}\,{}^n\boldsymbol{M}_n\,{}^n\boldsymbol{v}_n = {}^n\mathcal{F}_{n,T} \tag{8.2}$$

where

m_n is the mass of body B_n,

${}^n\boldsymbol{c}_n$ is the first moment of mass of body B_n about O_n,

${}^n\boldsymbol{J}_n$ is the second moment of mass (moment of inertia) of body B_n about O_n,

${}^n\boldsymbol{M}_n$ is the mass matrix of body B_n,

${}^n\boldsymbol{v}_n$ is the generalised velocity of body B_n,

${}^n\dot{\boldsymbol{v}}_n$ is the generalised acceleration of body B_n,

and ${}^n\mathcal{F}_{n,T}$ is the generalised total body force acting on body B_n.

8.3 Interbody geometric constraints

Equation (8.2) does not as yet describe a chain of bodies since it does not take into consideration the interbody constraints imposed by the joints. The interbody geometrical constraints and interbody forces must now be applied. Since multibody systems are being dealt with, all velocity and acceleration vectors are assumed to be relative to an inertial frame unless stated otherwise. An example of a vector that is not relative to an inertial frame is the vector representing the angular velocity of frame $\mathcal{F}_{n+1}$ relative to frame $\mathcal{F}_n$ which is written as $\underline{\omega}_{n+1,n}$.

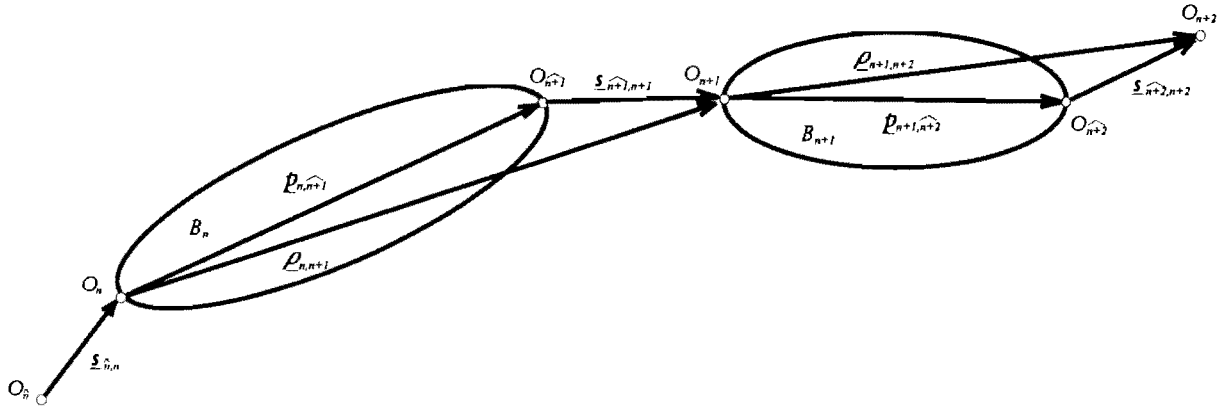


Figure 8.1 Two neighbouring bodies.

Consider figure 8.1. Let O_{n+1} be the point in body B_n at which body B_{n+1} is nominally attached. Since B_n is rigid then

$$\underline{\mathbf{v}}_{O_{\hat{n}+1}} = \underline{\mathbf{v}}_n + \underline{\omega}_n \times \underline{\boldsymbol{p}}_{n,\hat{n}+1}. \tag{8.3}$$

Note: if B_{n+1} is attached rigidly to B_n as in the case of rotational joints, then $O_{n\hat{+}1} \equiv O_{n+1}$. However, allowing for arbitrary translational motions between the bodies at their common joints the translational velocity of the body B_{n+1} can be written as

$$\underline{\mathbf{v}}_{n+1} \equiv \underline{\mathbf{v}}_{O_{n+1}} = \underline{\mathbf{v}}_{O_{n\hat{+}1}} + \underline{\mathbf{v}}_{n+1,n\hat{+}1} \tag{8.4}$$

where $\underline{\mathbf{v}}_{n+1,n\hat{+}1}$ is the velocity of O_{n+1} with respect to $O_{n\hat{+}1}$. Combining (8.3) and (8.4) leads to

$$\underline{\mathbf{v}}_{n+1} = \underline{\mathbf{v}}_n + \underline{\omega}_n \times \underline{\boldsymbol{p}}_{n,n\hat{+}1} + \underline{\mathbf{v}}_{n+1,n\hat{+}1}. \tag{8.5}$$

Expression (8.5) can be expressed in component form i.e.

$${}^{n+1}\mathbf{v}_{n+1} = {}^{n+1}\mathbf{C}_n\left({}^{n}\mathbf{v}_n - {}^{n}\boldsymbol{p}^{\times}_{n,n\hat{+}1}\ {}^{n}\omega_n\right) + {}^{n+1}\mathbf{v}_{n+1,n\hat{+}1}. \tag{8.6}$$

The angular velocities of the system of bodies are now considered. If rotation is permitted at $O_{n\hat{+}1}$ then

$$\underline{\omega}_{n+1} = \underline{\omega}_n + \underline{\omega}_{n+1,n\hat{+}1} \tag{8.7}$$

where $\underline{\omega}_{n+1,n\hat{+}1}$ is the angular velocity of B_{n+1} with respect to B_n about $O_{n\hat{+}1}$.
Expression (8.7) can be expressed in component form as

$${}^{n+1}\omega_{n+1} = {}^{n+1}\mathbf{C}_n\ {}^{n}\omega_n + {}^{n+1}\omega_{n+1,n\hat{+}1}. \tag{8.8}$$

The relationship between $\underline{\mathbf{v}}_{n+1,n\hat{+}1}$ and the displacement $\underline{\boldsymbol{s}}_{n\hat{+}1,n+1}$ shown in figure 8.1 is now found in vectorial terms:

$$\underline{\mathbf{v}}_{n+1,n\hat{+}1} = \dot{\underline{\boldsymbol{s}}}_{n\hat{+}1,n+1},$$

i.e. $\underline{\mathbf{v}}_{n+1,n\hat{+}1}$ equals the time derivative of $\underline{\boldsymbol{s}}_{n\hat{+}1,n+1}$ with respect to the inertial frame. Now denote time differentiation with respect to frame $\mathcal{F}_{n+1}$ by an over prime ' so that

$$\underline{\mathbf{v}}_{n+1,n\hat{+}1} = \underline{\boldsymbol{s}}'_{n\hat{+}1,n+1} + \underline{\omega}_{n+1} \times \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}. \tag{8.9}$$

Substituting for (8.9) and (8.7) into (8.5) yields

$$\begin{aligned} \underline{\mathbf{v}}_{n+1} &= \underline{\mathbf{v}}_n + \underline{\omega}_n \times \underline{\boldsymbol{p}}_{n,n\hat{+}1} + \underline{\boldsymbol{s}}'_{n\hat{+}1,n+1} + (\underline{\omega}_n + \underline{\omega}_{n+1,n\hat{+}1}) \times \underline{\boldsymbol{s}}_{n\hat{+}1,n+1} \\ &= \underline{\mathbf{v}}_n + \underline{\omega}_n \times (\underline{\boldsymbol{p}}_{n,n\hat{+}1} + \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}) + \underline{\boldsymbol{s}}'_{n\hat{+}1,n+1} + \underline{\omega}_{n+1,n\hat{+}1} \times \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}. \end{aligned} \tag{8.10}$$

Note that $\underline{\boldsymbol{\rho}}_{n,n+1} = \underline{\boldsymbol{p}}_{n,n\hat{+}1} + \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}$ so (8.10) can be written as

$$\underline{\mathbf{v}}_{n+1} = \underline{\mathbf{v}}_n + \underline{\omega}_n \times \underline{\boldsymbol{\rho}}_{n,n+1} + \underline{\boldsymbol{s}}'_{n\hat{+}1,n+1} + \underline{\omega}_{n+1,n\hat{+}1} \times \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}. \tag{8.11}$$

Expression (8.11) can be written in component form, i.e.

$${}^{n+1}\mathbf{v}_{n+1} = {}^{n+1}\mathbf{C}_n\left({}^{n}\mathbf{v}_n - {}^{n}\boldsymbol{\rho}^{\times}_{n,n+1}\ {}^{n}\omega_n\right) + {}^{n+1}\mathbf{w}_{n+1,n\hat{+}1} - {}^{n+1}\boldsymbol{s}^{\times}_{n\hat{+}1,n+1}\ {}^{n+1}\omega_{n+1,n\hat{+}1}. \tag{8.12}$$

where ${}^{n+1}\mathbf{w}_{n+1,n\hat{+}1}$ represents the velocity of ${}^{n+1}\boldsymbol{s}_{n\hat{+}1,n+1}$ as measured (as seen) in the frame $\mathcal{F}_{n+1}$.

i.e. ${}^{n+1}\mathbf{w}_{n+1,n\hat{+}1} = {}^{n+1}\dot{\boldsymbol{s}}_{n\hat{+}1,n+1} \equiv \mathcal{F}_{n+1}\underline{\boldsymbol{s}}'_{n\hat{+}1,n+1}$.

8.3.1 Recursive formulation of the generalised velocities and accelerations

The interbody translational velocity relationship (8.12) and the interbody angular velocity relationship (8.8) can be combined as follows:

$$\begin{pmatrix} {}^{n+1}\mathbf{v}_{n+1} \\ {}^{n+1}\omega_{n+1} \end{pmatrix} = \begin{bmatrix} {}^{n+1}\mathbf{C}_n & -{}^{n+1}\mathbf{C}_n\,{}^{n}\rho^{\times}_{n,n+1} \\ \mathbf{O} & {}^{n+1}\mathbf{C}_n \end{bmatrix} \begin{pmatrix} {}^{n}\mathbf{v}_n \\ {}^{n}\omega_n \end{pmatrix} + \begin{bmatrix} \mathbf{1} & -{}^{n+1}\mathbf{s}^{\times}_{n\hat{+}1,n+1} \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{pmatrix} {}^{n+1}\mathbf{w}_{n+1,n\hat{+}1} \\ {}^{n+1}\omega_{n+1,n\hat{+}1} \end{pmatrix}.$$

The above can be simply denoted by

$${}^{n+1}\boldsymbol{v}_{n+1} = {}^{n+1}\mathcal{T}_n\,{}^{n}\boldsymbol{v}_n + {}^{n+1}\mathcal{G}_{n+1,n+1}\,{}^{n+1}\boldsymbol{v}_{n+1,n\hat{+}1} \tag{8.13}$$

where

$${}^{n+1}\mathcal{T}_n = \begin{bmatrix} {}^{n+1}\mathbf{C}_n & -{}^{n+1}\mathbf{C}_n\,{}^{n}\rho^{\times}_{n,n+1} \\ \mathbf{O} & {}^{n+1}\mathbf{C}_n \end{bmatrix} \tag{8.14}$$

and

$${}^{n+1}\mathcal{G}_{n+1,n+1} = \begin{bmatrix} \mathbf{1} & -{}^{n+1}\mathbf{s}^{\times}_{n\hat{+}1,n+1} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}. \tag{8.15}$$

The interbody velocity can be expressed as a function of the joint rates and a projection matrix.

$${}^{n+1}\boldsymbol{v}_{n+1,n\hat{+}1} = \begin{pmatrix} {}^{n+1}\mathbf{w}_{n+1,n\hat{+}1} \\ {}^{n+1}\omega_{n+1,n\hat{+}1} \end{pmatrix} = \mathbf{P}_{n+1}\,\dot{\beta}_{n+1} \tag{8.16}$$

where $\dot{\beta}_{n+1}$ is the rate of the n+1 joint, and $\mathbf{P}_{n+1}$ is the projection matrix mapping the rate of the n+1 joint, $\dot{\beta}_{n+1}$, on to a specified axis or axes (multi dof joints). For a revolute joint with its axis of revolution aligned with the 2 axis of the n+1 frame the projection matrix $\mathbf{P}_{n+1}$ is given by

$$\mathbf{P}_{n+1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

A detailed definition and treatment of projection matrices is presented by Hughes, Sincarsin and D'Eleuterio [1988,1989].

Differentiating (8.13) with respect to time gives an interbody relationship for the generalised acceleration

$${}^{n+1}\dot{\boldsymbol{v}}_{n+1} = {}^{n+1}\mathcal{T}_n\,{}^{n}\dot{\boldsymbol{v}}_n + {}^{n+1}\dot{\mathcal{T}}_n\,{}^{n}\boldsymbol{v}_n + {}^{n+1}\dot{\mathcal{G}}_{n+1,n+1}\,{}^{n+1}\boldsymbol{v}_{n+1,n\hat{+}1} + {}^{n+1}\mathcal{G}_{n+1,n+1}\,{}^{n+1}\dot{\boldsymbol{v}}_{n+1,n\hat{+}1}. \tag{8.17}$$

To find the time derivative of the interbody transformation matrix ${}^{n+1}\mathcal{T}_n$ first decompose the matrix as follows:

$${}^{n+1}\mathcal{T}_n = {}^{n+1}\mathbf{C}_n\,{}^{n+1}\mathbf{R}_n \tag{8.18}$$

where

$$ {}^{n+1}\mathbf{C}_n = \begin{bmatrix} {}^{n+1}\mathbf{C}_n & \mathbf{O} \\ \mathbf{O} & {}^{n+1}\mathbf{C}_n \end{bmatrix} $$

and

$$ {}^{n+1}\mathbf{R}_n = \begin{bmatrix} \mathbf{1} & -{}^{n}\boldsymbol{\rho}^{\times}_{n,n+1} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}. \tag{8.19} $$

Differentiating (8.18) gives

$$ {}^{n+1}\dot{\mathcal{T}}_n = {}^{n+1}\dot{\mathbf{C}}_n \, {}^{n+1}\mathbf{R}_n + {}^{n+1}\mathbf{C}_n \, {}^{n+1}\dot{\mathbf{R}}_n \tag{8.20} $$

where

$$ {}^{n+1}\dot{\mathbf{C}}_n = \begin{bmatrix} {}^{n+1}\dot{\mathbf{C}}_n & \mathbf{O} \\ \mathbf{O} & {}^{n+1}\dot{\mathbf{C}}_n \end{bmatrix}. $$

Moreover

$$ \begin{aligned} {}^{n+1}\dot{\mathbf{C}}_n &= -{}^{n+1}\boldsymbol{\omega}^{\times}_{n+1,n} \, {}^{n+1}\mathbf{C}_n \\ &= -{}^{n+1}\boldsymbol{\omega}^{\times}_{n+1,n\hat{+}1} \, {}^{n+1}\mathbf{C}_n \end{aligned} $$

given (8.8) and the fact that

$$ {}^{n+1}\boldsymbol{\omega}_{n+1} = {}^{n+1}\mathbf{C}_n \, {}^{n}\boldsymbol{\omega}_n + {}^{n+1}\boldsymbol{\omega}_{n+1,n}; $$

hence

$$ {}^{n+1}\dot{\mathbf{C}}_n = {}^{n+1}\boldsymbol{\Omega}^{\times^T}_{n+1,n} \, {}^{n+1}\mathbf{C}_n \tag{8.21} $$

where

$$ {}^{n+1}\boldsymbol{\Omega}^{\times}_{n+1,n} = \begin{bmatrix} {}^{n+1}\boldsymbol{\omega}^{\times}_{n+1,n\hat{+}1} & \mathbf{O} \\ \mathbf{O} & {}^{n+1}\boldsymbol{\omega}^{\times}_{n+1,n\hat{+}1} \end{bmatrix}. $$

The first term on the right hand side of (8.20) can be written as

$$ {}^{n+1}\dot{\mathbf{C}}_n \, {}^{n+1}\mathbf{R}_n = {}^{n+1}\boldsymbol{\Omega}^{\times^T}_{n+1,n} \, {}^{n+1}\mathbf{C}_n \, {}^{n+1}\mathbf{R}_n = {}^{n+1}\boldsymbol{\Omega}^{\times^T}_{n+1,n} \, {}^{n+1}\mathcal{T}_n . \tag{8.22} $$

Thus, an expression for the first term on the right hand side of (8.20) has been derived that does not involve derivatives.

Differentiating (8.19) yields

$$ {}^{n+1}\dot{\mathbf{R}}_n = \begin{bmatrix} \mathbf{O} & -{}^{n}\dot{\boldsymbol{\rho}}^{\times}_{n,n+1} \\ \mathbf{O} & \mathbf{O} \end{bmatrix} $$

noting that $\underline{\boldsymbol{\rho}}_{n,n+1} = \underline{\boldsymbol{p}}_{n,n\hat{+}1} + \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}$

so that

$$ \begin{aligned} \dot{\underline{\boldsymbol{\rho}}}_{n,n+1} &= \dot{\underline{\boldsymbol{p}}}_{n,n\hat{+}1} + \dot{\underline{\boldsymbol{s}}}_{n\hat{+}1,n+1} \\ &= \overset{o}{\underline{\boldsymbol{p}}}_{n,n\hat{+}1} + \underline{\boldsymbol{\omega}}_n \times \underline{\boldsymbol{p}}_{n,n\hat{+}1} + \dot{\underline{\boldsymbol{s}}}_{n\hat{+}1,n+1} + \underline{\boldsymbol{\omega}}_{n+1} \times \underline{\boldsymbol{s}}_{n\hat{+}1,n+1}. \end{aligned} \tag{8.23} $$

But $\overset{o}{\underline{p}}_{n,n\hat{+}1} = \underline{0}$ since $\underline{p}_{n,n\hat{+}1}$ is a constant vector in frame $\mathcal{F}_n$. Recall that $\dot{}$ denotes time differentiation with respect to the inertial frame $\mathcal{F}_I$, and ^{o} denotes time differentiation with respect to $\square_n$ and $'$ denotes time differentiation with respect to $\mathcal{F}_{n+1}$.

However the derivative of $\underline{\rho}_{n,n+1}$ with respect to $\mathcal{F}_I$ can also be written as

$$\dot{\underline{\rho}}_{n,n+1} = \overset{o}{\underline{\rho}}_{n,n+1} + \underline{\omega}_n \times \underline{\rho}_{n,n+1}. \tag{8.24}$$

Equate (8.23) and (8.24) and note that $\overset{o}{\underline{p}}_{n,n\hat{+}1} = \underline{0}$ to obtain

$$\begin{aligned}
\overset{o}{\underline{\rho}}_{n,n+1} &= \underline{\omega}_n \times \underline{p}_{n,n\hat{+}1} + \underline{s}'_{n\hat{+}1,n+1} + \underline{\omega}_{n+1} \times \underline{s}_{n\hat{+}1,n+1} - \underline{\omega}_n \times \underline{\rho}_{n,n+1} \\
&= \underline{\omega}_n \times \underline{p}_{n,n\hat{+}1} + \underline{\omega}_{n+1} \times \underline{s}_{n\hat{+}1,n+1} - \underline{\omega}_n \times \underline{p}_{n,n\hat{+}1} - \underline{\omega}_n \times \underline{s}_{n\hat{+}1,n+1} + \underline{s}'_{n\hat{+}1,n+1} \\
&= \underline{s}'_{n\hat{+}1,n+1} + (\underline{\omega}_{n+1} - \underline{\omega}_n) \times \underline{s}_{n\hat{+}1,n+1}.
\end{aligned} \tag{8.25}$$

Given (8.8) the above can be written as

$$\overset{o}{\underline{\rho}}_{n,n+1} = \underline{s}'_{n\hat{+}1,n+1} + \underline{\omega}_{n+1,n\hat{+}1} \times \underline{s}_{n\hat{+}1,n+1}.$$

In component form this is

$${}^n\dot{\rho}_{n,n+1} = {}^n\mathbf{C}_{n+1}\left({}^{n+1}\mathbf{w}_{n\hat{+}1,n+1} - {}^{n+1}\mathbf{s}^{\times}_{n\hat{+}1,n+1}\,{}^{n+1}\omega_{n+1,n\hat{+}1}\right)$$

remembering that ${}^n\dot{\rho}_{n,n+1} = \mathcal{F}_n\overset{\circ}{\underline{\rho}}_{n,n+1}$.

The second term on the right hand side of (8.20) is

$$\begin{aligned}
{}^{n+1}\mathbf{C}_n\,{}^{n+1}\dot{\mathbf{R}}_n &= \begin{bmatrix} \mathbf{O} & -{}^{n+1}\mathbf{w}^{\times}_{n+1,n\hat{+}1}\,{}^{n+1}\mathbf{C}_n + \left({}^{n+1}\mathbf{s}^{\times}_{n\hat{+}1,n+1}\,{}^{n+1}\omega_{n+1,n\hat{+}1}\right)^{\times}{}^{n+1}\mathbf{C}_n \\ \mathbf{O} & \mathbf{O} \end{bmatrix} \\
&= {}^{n+1}\mathbf{V}^{\times T}_{n+1,n}\,{}^{n+1}\mathcal{J}_n
\end{aligned} \tag{8.26}$$

where

$${}^{n+1}\mathbf{V}^{\times}_{n+1,n} = \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ {}^{n+1}\mathbf{w}^{\times}_{n+1,n\hat{+}1} - \left({}^{n+1}\mathbf{s}^{\times}_{n\hat{+}1,n+1}\,{}^{n+1}\omega_{n+1,n\hat{+}1}\right)^{\times} & \mathbf{O} \end{bmatrix}.$$

Thus, an expression for the 2nd term on the right hand side of (8.20) has been derived that does not involve derivatives.

Combining (8.22) and (8.26) gives

$$\begin{aligned}
{}^{n+1}\dot{\mathcal{J}}_n &= \left({}^{n+1}\Omega^{\times T}_{n+1,n} + {}^{n+1}\mathbf{V}^{\times T}_{n+1,n}\right){}^{n+1}\mathcal{J}_n \\
&= \left({}^{n+1}\Omega^{\times}_{n+1,n} + {}^{n+1}\mathbf{V}^{\times}_{n+1,n}\right)^T {}^{n+1}\mathcal{J}_n \\
&= {}^{n+1}\mathcal{G}_{n+1,n+1}\left({}^{n+1}v^{\otimes}_{n+1,n\hat{+}1}\right)^T {}^{n+1}\mathcal{G}^{-1}_{n+1,n+1}\,{}^{n+1}\mathcal{J}_n
\end{aligned}$$

where

$${}^{n+1}v^{\otimes}_{n+1,n\hat{+}1} = \begin{bmatrix} {}^{n+1}\omega^{\times}_{n\hat{+}1,n+1} & \mathbf{O} \\ {}^{n+1}\mathbf{w}^{\times}_{n\hat{+}1,n+1} & {}^{n+1}\omega^{\times}_{n\hat{+}1,n+1} \end{bmatrix}.$$

A simple relationship exists for ${}^{n+1}\dot{\mathcal{G}}_{n+1,n+1}$. Since ${}^{n+1}\mathbf{w}_{n\hat{+}1,n+1} = {}^{n+1}\dot{\mathbf{s}}_{n\hat{+}1,n+1}$ then it follows from (8.15) that

$$ {}^{n+1}\dot{\mathcal{G}}_{n+1,n+1} = \begin{bmatrix} \mathbf{O} & -{}^{n+1}\mathbf{w}^{\times}_{n\hat{+}1,n+1} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}. $$

8.4 Interbody force constraints

So far each body has been taken as an independent entity from the view point of interbody forces. The constraint that the generalised force on body n from body n+1 is equal and opposite to the generalised force on body n+1 due to body n is now applied.

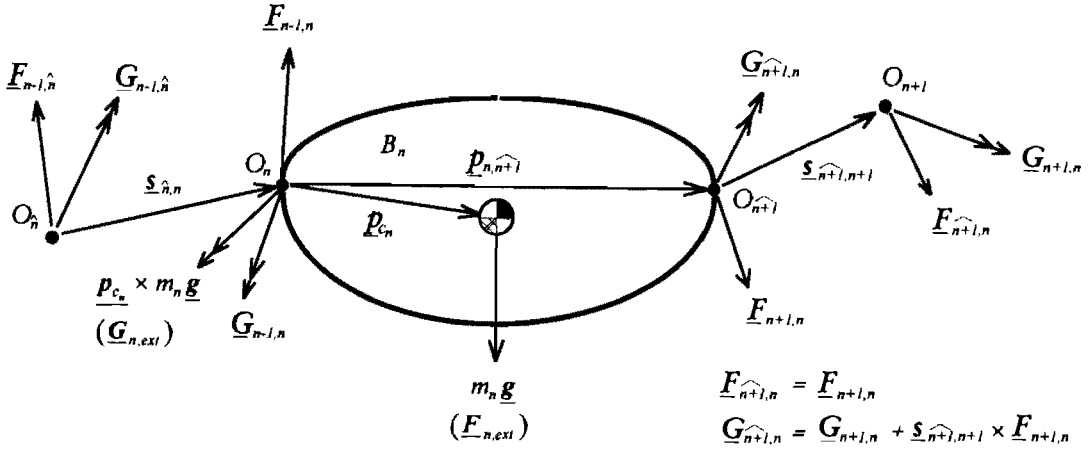


Figure 8.2 Interbody forces and torques.

Referring to figure 8.2, let $\underline{F}_{n+1,n}$ be the force on body B_n due to B_{n+1}, and let $\underline{G}_{n+1,n}$ be the torque about O_{n+1} on B_n due to B_{n+1}. Similarly, $\underline{F}_{n-1,n}$ is the force on B_n due to B_{n-1}, and $\underline{G}_{n-1,n}$ is the torque about O_n on B_n due to B_{n-1}. In addition, let $\underline{F}_{n\hat{+}1,n}$ and $\underline{G}_{n\hat{+}1,n}$ be the force and torque acting on B_n due to B_{n+1} at $O_{n\hat{+}1}$.

Now the total force $\underline{F}_{n,T}$ acting on B_n at O_n is

$$ \underline{F}_{n,T} = \underline{F}_{n-1,n} + \underline{F}_{n\hat{+}1,n} + \underline{F}_{n,ext} \tag{8.27} $$

and the total torque $\underline{G}_{n,T}$ acting on B_n at O_n is

$$ \underline{G}_{n,T} = \underline{G}_{n-1,n} + \underline{G}_{n\hat{+}1,n} + \underline{p}_{n,n\hat{+}1} \times \underline{F}_{n+1,n} + \underline{G}_{n,ext}. \tag{8.28} $$

The terms $\underline{F}_{n,ext}$ and $\underline{G}_{n,ext}$ are the external force and torque on B_n, i.e. the force and torque not attributable to interbody effects. The external force and torque due to gravity can be written as

$$ \underline{F}_{n,ext} = m_n \underline{g}, $$

$$ \underline{G}_{n,ext} = \underline{p}_{c_n} \times m_n \underline{g} $$

where $\underline{p}_{c_n}$ is the position of the centre of mass of B_n relative to O_n.

Note that the translational velocity associated with the prismatic joint between B_n and B_{n+1} is assumed to be measured at O_{n+1}, whereas the angular velocity associated with the revolute joint between B_n and B_{n+1} is assumed to be measured at $O_{n\hat{+}1}$. Hence the force and torque initiating each of the above motions is taken to be applied at the same point at which its respective

resultant velocity is measured. It is therefore necessary to manipulate expressions (8.27) and (8.28) so that they are consistent with section 8.3. Note that

$$\underline{F}_{n\hat{+}1,n} = \underline{F}_{n+1,n}, \tag{8.29}$$

$$\underline{G}_{n-1,\hat{n}} = \underline{G}_{n-1,n} + \underline{s}_{\hat{n},n} \times \underline{F}_{n-1,n}, \tag{8.30}$$

and

$$\underline{F}_{n,n+1} = -\underline{F}_{n+1,n}, \quad \underline{F}_{\hat{n},n+1} = -\underline{F}_{n+1,\hat{n}},$$

$$\underline{G}_{n,n+1} = -\underline{G}_{n+1,n}, \quad \underline{G}_{\hat{n},n+1} = -\underline{G}_{n+1,\hat{n}}.$$

Substituting for expressions (8.29) and (8.30) in to (8.27) and (8.28) yields

$$\underline{F}_{n,T} = -\underline{F}_{n,n+1} + \underline{F}_{n-1,n} + \underline{F}_{n,ext}, \tag{8.31}$$

$$\underline{G}_{n,T} = -\underline{G}_{n,n\hat{+}1} + \underline{G}_{n-1,\hat{n}} - \underline{\rho}_{n,n+1} \times \underline{F}_{n,n+1} + \underline{s}_{n\hat{+}1,n+1} \times \underline{F}_{n,n+1} - \underline{s}_{\hat{n},n} \times \underline{F}_{n-1,n} + \underline{G}_{n,ext}. \tag{8.32}$$

Expressions (8.31) and (8.32) can be written in component form and combined as follows:

$$\begin{pmatrix} {}^{n}\boldsymbol{F}_{n,T} \\ {}^{n}\boldsymbol{G}_{n,T} \end{pmatrix} = -\begin{bmatrix} {}^{n}\mathbf{C}_{n+1} & \mathbf{O} \\ {}^{n}\boldsymbol{\rho}^{\times}_{n,n+1}\,{}^{n}\mathbf{C}_{n+1} - {}^{n}\mathbf{C}_{n+1}\,{}^{n+1}\boldsymbol{s}^{\times}_{n\hat{+}1,n+1} & {}^{n}\boldsymbol{C}_{n+1} \end{bmatrix} \begin{pmatrix} {}^{n+1}\boldsymbol{F}_{n,n+1} \\ {}^{n+1}\boldsymbol{G}_{n,n\hat{+}1} \end{pmatrix}$$
$$+ \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ -{}^{n}\boldsymbol{s}^{\times}_{\hat{n},n} & \mathbf{1} \end{bmatrix} \begin{pmatrix} {}^{n}\boldsymbol{F}_{n-1,n} \\ {}^{n}\boldsymbol{G}_{n-1,\hat{n}} \end{pmatrix} + \begin{pmatrix} {}^{n}\boldsymbol{F}_{n,ext} \\ {}^{n}\boldsymbol{G}_{n,ext} \end{pmatrix}.$$

The external force and torque due to gravity can be written as

$${}^{n}\boldsymbol{F}_{n,ext} = m_n\,{}^{n}\boldsymbol{g},$$

$${}^{n}\boldsymbol{G}_{n,ext} = {}^{n}\boldsymbol{c}_n^{\times}\,{}^{n}\boldsymbol{g} \qquad \text{where } {}^{n}\boldsymbol{c}_n = m_m\,{}^{n}\boldsymbol{p}_{c_n}.$$

The final expression for the total generalised force on B_n is

$${}^{n}\mathcal{F}_{n,T} = -{}^{n+1}\mathcal{J}_n^{T}\,{}^{n+1}\mathcal{G}^{-T}_{n+1,n+1}\,{}^{n+1}\mathcal{F}_{n,n+1} + {}^{n}\mathcal{G}^{-T}_{n,n}\,{}^{n}\mathcal{F}_{n-1,n} + {}^{n}\mathcal{F}_{n,ext}.$$

8.4.1 Recursive formulation of the generalised forces

Noting the final expression for the total generalised force on B_n written above, the generalised force on body B_n due to B_{n-1} can be written as

$${}^{n}\mathcal{F}_{n-1,n} = \mathcal{G}^{T}_{n,n}\left({}^{n+1}\mathcal{J}_n^{T}\,{}^{n+1}\mathcal{G}^{-T}_{n+1,n+1}\,{}^{n+1}\mathcal{F}_{n,n+1} + {}^{n}\mathcal{F}_{n,T} - {}^{n}\mathcal{F}_{n,ext} \right). \tag{8.33}$$

Note that ${}^{n}\mathcal{F}_{n-1,n}$ cannot be calculated without ${}^{n+1}\mathcal{F}_{n,n+1}$; therefore (8.33) can be thought of as a recursive formula for the joint forces from body B_N due to B_0.

8.5 Global formulation of the motion equations

The global formulation of the motion equations is now developed. The objective is to combine the motion equations derived so far in order to describe the motion of the system of bodies as a whole.

8.5.1 Global formulation of the generalised velocities

Recall from (8.13) that the velocity kinematics of the n[th] body in the chain is given by

$$ {}^{n+1}v_{n+1} = {}^{n+1}\mathcal{T}_n \, {}^{n}v_n + {}^{n+1}\mathcal{G}_{n+1,n+1} \, {}^{n+1}v_{n+1,n+\hat{1}} . \tag{8.34} $$

Given that the generalised velocity of the body B_0 is zero, i.e. fixed with respect to the inertial frame, the generalised velocity of the body B_0 can be expressed as

$$ {}^{0}v_0 = 0 . $$

Using (8.34) the rest of the generalised velocities moving outward from body B_0 can be written as

$$
\begin{aligned}
{}^{1}v_1 &= {}^{1}\mathcal{G}_{1,1} \, {}^{1}v_{1,\hat{1}} , \\
{}^{2}v_2 &= {}^{2}\mathcal{T}_1 \, {}^{1}v_1 + {}^{2}\mathcal{G}_{2,2} \, {}^{2}v_{2,\hat{2}} \\
&= {}^{2}\mathcal{T}_1 \, {}^{1}\mathcal{G}_{1,1} \, {}^{1}v_{1,\hat{1}} + {}^{2}\mathcal{G}_{2,2} \, {}^{2}v_{2,\hat{2}} , \\
{}^{3}v_3 &= {}^{3}\mathcal{T}_2 \, {}^{2}v_2 + {}^{3}\mathcal{G}_{3,3} \, {}^{3}v_{3,\hat{3}} \\
&= {}^{3}\mathcal{T}_2 \left({}^{2}\mathcal{T}_1 \, {}^{1}\mathcal{G}_{1,1} \, {}^{1}v_{1,\hat{1}} + {}^{2}\mathcal{G}_{2,2} \, {}^{2}v_{2,\hat{2}} \right) + {}^{3}\mathcal{G}_{3,3} \, {}^{3}v_{3,\hat{3}} \\
&= {}^{3}\mathcal{T}_1 \, {}^{1}\mathcal{G}_{1,1} \, {}^{1}v_{1,\hat{1}} + {}^{3}\mathcal{T}_2 \, {}^{2}\mathcal{G}_{2,2} \, {}^{2}v_{2,\hat{2}} + {}^{3}\mathcal{G}_{3,3} \, {}^{3}v_{3,\hat{3}} , \\
&\vdots \\
&\vdots \\
&\vdots \\
{}^{N}v_N &= {}^{N}\mathcal{T}_1 \, {}^{1}\mathcal{G}_{1,1} \, {}^{1}v_{1,\hat{1}} \\
&\quad + {}^{N}\mathcal{T}_2 \, {}^{2}\mathcal{G}_{2,2} \, {}^{2}v_{2,\hat{2}} \\
&\quad + {}^{N}\mathcal{T}_3 \, {}^{3}\mathcal{G}_{3,3} \, {}^{3}v_{3,\hat{3}} \\
&\quad + \ \ldots \\
&\quad + {}^{N}\mathcal{G}_{N,N} \, {}^{N}v_{N,\hat{N}} .
\end{aligned}
$$

The system of generalised velocities or global velocity for the entire system can be written as

$$
\begin{pmatrix} {}^{1}v_1 \\ {}^{2}v_2 \\ {}^{3}v_3 \\ \vdots \\ {}^{N}v_N \end{pmatrix} =
\begin{bmatrix} \mathbf{1} & & & & \mathbf{O} \\ {}^{2}\mathcal{T}_1 & \mathbf{1} & & & \\ {}^{3}\mathcal{T}_1 & {}^{3}\mathcal{T}_2 & \mathbf{1} & & \\ \vdots & \vdots & \vdots & \ddots & \\ {}^{N}\mathcal{T}_1 & {}^{N}\mathcal{T}_2 & {}^{N}\mathcal{T}_3 & \ldots & \mathbf{1} \end{bmatrix}
\begin{bmatrix} {}^{1}\mathcal{G}_{1,1} & & & & \mathbf{O} \\ & {}^{2}\mathcal{G}_{2,2} & & & \\ & & {}^{3}\mathcal{G}_{3,3} & & \\ & & & \ddots & \\ \mathbf{O} & & & & {}^{N}\mathcal{G}_{N,N} \end{bmatrix}
\begin{pmatrix} {}^{1}v_{1,\hat{1}} \\ {}^{2}v_{2,\hat{2}} \\ {}^{3}v_{3,\hat{3}} \\ \vdots \\ {}^{N}v_{N,\hat{N}} \end{pmatrix} .
$$

The above can be simply denoted by

$$ v = \mathcal{T} \mathcal{G} \hat{v} . \tag{8.35} $$

Recall from section 8.3.1 that ${}^{n+1}v_{n+1,n+\hat{1}} = \mathbf{P}_{n+1} \dot{\beta}_{n+1}$; thus the assembled system of interbody velocities can be written as $\hat{v} = \mathbf{P}\dot{\beta}$. Expression (8.35) can now be written as

$$ v = \mathcal{T} \mathcal{G} \mathbf{P} \dot{\beta} \tag{8.36} $$

where $\dot{\beta}$ and $\mathbf{P}$ are given by

$$\dot{\beta} = \begin{pmatrix} \dot{\beta}_1 \\ \dot{\beta}_2 \\ \dot{\beta}_3 \\ \vdots \\ \dot{\beta}_N \end{pmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & & & & \mathbf{O} \\ & \mathbf{P}_2 & & & \\ & & \mathbf{P}_3 & & \\ & & & \ddots & \\ \mathbf{O} & & & & \mathbf{P}_N \end{bmatrix}.$$

8.5.2 Global formulation of the generalised forces for a open chain

Recall from section 8.4 the generalised total body force

$${}^{n}\mathcal{F}_{n,T} = -{}^{n+1}\mathcal{J}_n^T \, {}^{n+1}\mathcal{G}_{n+1,n+1}^{-T} \, {}^{n+1}\mathcal{F}_{n,n+1} + {}^{n}\mathcal{G}_{n,n}^{-T} \, {}^{n}\mathcal{F}_{n-1,n} + {}^{n}\mathcal{F}_{n,ext}. \tag{8.37}$$

For an open looped chain the forces exerted by link N onto link N+1 are zero, i.e.

$${}^{N+1}\mathcal{F}_{N,N+1} = 0.$$

Using (8.37) it is possible to write the rest of the generalised total body forces moving inward from body B_{N+1} as

$$\begin{aligned}
{}^{N}\mathcal{F}_{N,T} &= {}^{N}\mathcal{G}_{N,N}^{-T} \, {}^{N}\mathcal{F}_{N-1,N} + {}^{N}\mathcal{F}_{N,ext}, \\
{}^{N-1}\mathcal{F}_{N-1,T} &= -{}^{N}\mathcal{J}_{N-1}^{T} \, {}^{N}\mathcal{G}_{N,N}^{-T} \, {}^{N}\mathcal{F}_{N-1,N} + {}^{N-1}\mathcal{G}_{N-1,N-1}^{-T} \, {}^{N-1}\mathcal{F}_{N-2,N-1} + {}^{N-1}\mathcal{F}_{N-1,ext}, \\
&\vdots \\
&\vdots \\
&\vdots \\
{}^{1}\mathcal{F}_{1,T} &= -{}^{2}\mathcal{J}_{1}^{T} \, {}^{2}\mathcal{G}_{2,2}^{-T} \, {}^{2}\mathcal{F}_{1,2} + {}^{1}\mathcal{G}_{1,1}^{-T} \, {}^{1}\mathcal{F}_{0,1} + {}^{1}\mathcal{F}_{1,ext}.
\end{aligned}$$

The assembled (global) system of generalised total body forces can be written as

$$\begin{pmatrix} {}^{1}\mathcal{F}_{1,T} \\ {}^{2}\mathcal{F}_{2,T} \\ {}^{3}\mathcal{F}_{3,T} \\ \vdots \\ {}^{N}\mathcal{F}_{N,T} \end{pmatrix} = \begin{bmatrix} \mathbf{1} & -{}^{2}\mathcal{J}_{1}^{T} & & & \mathbf{O} \\ & \mathbf{1} & -{}^{3}\mathcal{J}_{2}^{T} & & \\ & & \ddots & \ddots & \\ & & & \mathbf{1} & -{}^{N}\mathcal{J}_{N-1}^{T} \\ \mathbf{O} & & & & \mathbf{1} \end{bmatrix} \begin{bmatrix} {}^{1}\mathcal{G}_{1,1}^{-1} & & & & \mathbf{O} \\ & {}^{2}\mathcal{G}_{2,2}^{-1} & & & \\ & & {}^{3}\mathcal{G}_{3,3}^{-1} & & \\ & & & \ddots & \\ \mathbf{O} & & & & {}^{N}\mathcal{G}_{N,N}^{-1} \end{bmatrix} \begin{pmatrix} {}^{1}\mathcal{F}_{0,1} \\ {}^{2}\mathcal{F}_{1,2} \\ {}^{3}\mathcal{F}_{2,3} \\ \vdots \\ {}^{N}\mathcal{F}_{N-1,N} \end{pmatrix} + \begin{pmatrix} {}^{1}\mathcal{F}_{1,ext} \\ {}^{2}\mathcal{F}_{2,ext} \\ {}^{3}\mathcal{F}_{3,ext} \\ \vdots \\ {}^{N}\mathcal{F}_{N,ext} \end{pmatrix}. \tag{8.38}$$

Expression (8.38) can be simply denoted by

$$\mathcal{F}_T = \mathcal{J}^{-T} \mathcal{G}^{-T} \mathcal{F}_{Int} + \mathcal{F}_{Ext}. \tag{8.39}$$

The interbody forces ${}^{n}\mathcal{F}_{n-1,n}$ can be broken down into the control forces ${}^{n}\mathcal{F}_{C}$ (forces due to actuators) and the constraint forces ${}^{n}\mathcal{F}_{n,con}$ (forces acting on body B_n due to reactions with body B_{n-1}).

$${}^{n}\mathcal{F}_{n,n-1} = \mathbf{P}_n \, {}^{n}\mathcal{F}_{C} + \mathbf{Q}_n \, {}^{n}\mathcal{F}_{n,con}.$$

$\mathbf{Q}_n$ is the projection matrix mapping the constraint forces ${}^n\mathcal{F}_{n,con}$ on to specified axes. For a revolute joint with its axis of revolution aligned with the 2 axis of the n+1 frame the projection matrix $\mathbf{Q}_n$ is given by

$$\mathbf{Q}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For a detailed definition and treatment of projection matrices refer to Hughes, Sincarsin and D'Eleuterio [1988,1989]. The projection matrices $\mathbf{P}_n$ and $\mathbf{Q}_n$ have the following properties

$$\mathbf{P}_n^T\mathbf{P}_n = \mathbf{1}_{n,p},\ \mathbf{P}_n^T\mathbf{Q}_n = \mathbf{O},\ \mathbf{Q}_n^T\mathbf{P}_n = \mathbf{O},\ \mathbf{Q}_n^T\mathbf{Q}_n = \mathbf{1}_{n,q}.$$

Note that $\mathbf{1}_{n,p}$ and $\mathbf{1}_{n,q}$ will not necessarily be identity matrices. The interbody forces can be assembled into a system of generalised interbody forces as follows:

$$\mathcal{F}_{Int} = \mathbf{P}\mathcal{F}_C + \mathbf{Q}\mathcal{F}_{Con}$$

where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & & & \mathbf{O} \\ & \mathbf{Q}_2 & & \\ & & \mathbf{Q}_3 & \\ & & & \ddots \\ \mathbf{O} & & & \mathbf{Q}_N \end{bmatrix}.$$

Also

$$\mathbf{P}^T\mathbf{P} = \mathbf{1}_p,\ \mathbf{P}^T\mathbf{Q} = \mathbf{O},\ \mathbf{Q}^T\mathbf{P} = \mathbf{O},\ \mathbf{Q}^T\mathbf{Q} = \mathbf{1}_q.$$

8.5.3 Global constrained joint motion equations

The global equations are summarised below

$$\mathbf{M}\dot{v} = \mathcal{F}_T - v^{\otimes}\mathbf{M}v, \tag{8.40}$$

$$v = \mathcal{T}\mathcal{G}\mathbf{P}\dot{\beta}, \tag{8.41}$$

$$\mathcal{F}_T = \mathcal{T}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} + \mathcal{F}_{Ext}, \tag{8.42}$$

$$\mathcal{F}_{Int} = \mathbf{P}\mathcal{F}_C + \mathbf{Q}\mathcal{F}_{Con}. \tag{8.43}$$

Differentiating (8.41) with respect to time yields

$$\dot{v} = \mathcal{T}\mathcal{G}\mathbf{P}\ddot{\beta} + \left(\dot{\mathcal{T}}\mathcal{G}\mathbf{P} + \mathcal{T}\dot{\mathcal{G}}\mathbf{P} + \mathcal{T}\mathcal{G}\dot{\mathbf{P}}\right)\dot{\beta} \tag{8.44}$$

where

$$\dot{\mathcal{T}} = -\mathcal{T}\dot{\mathcal{T}}^{-1}\mathcal{T}.$$

Substituting for (8.44) into (8.40) leads to

$$\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{F}_T - v^{\otimes}\mathbf{M}v - \mathbf{M}\left(\dot{\mathcal{J}}\mathcal{G}\mathbf{P} + \mathcal{J}\dot{\mathcal{G}}\mathbf{P} + \mathcal{J}\mathcal{G}\dot{\mathbf{P}}\right)\dot{\beta}. \tag{8.45}$$

For convenience define $\mathcal{F}_{Non} = -v^{\otimes}\mathbf{M}v - \mathbf{M}\left(\dot{\mathcal{J}}\mathcal{G}\mathbf{P} + \mathcal{J}\dot{\mathcal{G}}\mathbf{P} + \mathcal{J}\mathcal{G}\dot{\mathbf{P}}\right)\dot{\beta}$. Substituting for (8.42) and $\mathcal{F}_{Non}$ into (8.45) yields

$$\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{J}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} + \mathcal{F}_{Ext} + \mathcal{F}_{Non}. \tag{8.46}$$

Pre multiplying (8.46) by $\mathbf{P}^T\mathcal{G}^T\mathcal{J}^T$ gives

$$\mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{P}^T\mathcal{F}_{Int} + \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}. \tag{8.47}$$

The constraint forces can be eliminated by substituting (8.43) to get

$$\mathbf{P}^T\hat{\mathcal{G}}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{P}^T\mathbf{P}\mathcal{F}_C + \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}. \tag{8.48}$$

The global constrained joint motion equation can now be written as

$$\mathbf{M}_{\beta\beta}\ddot{\beta} = \mathbf{1}_P\mathcal{F}_C + \mathcal{F}_E + \mathcal{F}_{Non,\beta} \tag{8.49}$$

where

$$\mathbf{M}_{\beta\beta} = \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P},$$
$$\mathcal{F}_E = \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext},$$
$$\mathcal{F}_{Non,\beta} = \mathbf{P}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}.$$

8.5.4 Calculation of the constraint forces

Premultiplying (8.46) by $\mathcal{G}^T\mathcal{J}^T$ forms

$$\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{F}_{Int} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}. \tag{8.50}$$

Substituting for (8.43) gives

$$\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{P}\mathcal{F}_C + \mathbf{Q}\mathcal{F}_{Con} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}. \tag{8.51}$$

Premultiplying (8.51) by $\mathbf{Q}^T$ to eliminate the open loop control torques $\mathcal{F}_C$ by virtue of the property of the projection matrices $\mathbf{Q}^T\mathbf{P} = \mathbf{O}$ gives

$$\mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{Q}^T\mathbf{Q}\mathcal{F}_{Con} + \mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non}. \tag{8.52}$$

The global constraint forces are then given by

$$\mathcal{F}_{Con} = \mathbf{1}_q\mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\left(\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} - \mathcal{F}_{Ext} - \mathcal{F}_{Non}\right).$$

The sets of equations needed for the dynamic analysis of the mechanism have now been derived. These equations are used in the next chapter to formulate the inverse dynamics of the closed looped mechanisms.

Chapter 9

Dynamic analysis of the mechanism

9.1 Introduction

The dynamics of constrained multi-body systems have been extensively studied in recent years: Wittenburg [1977], Luh and Zheng [1985], Kleinfinger and Khalil [1986], Nakamura and Ghodoussi [1988], Murray and Lovell [1989], García de Jalón and Bayo [1994], Shabana [1994]. Wittenburg [1977] introduced the notion of a reduced system, which is an open-chain mechanism obtained from a closed-chain mechanism by 'cutting' each kinematic loop in the system at an unactuated joint to produce a chain with a tree structure, thus removing the physical constraints. The equations of motion for the closed-chain mechanism are derived from the reduced system dynamic model by replacing the removed constraints according to a standard Lagrange multiplier approach.

Luh and Zheng [1985] formalised this methodology for closed-chain robotic manipulators and developed a two step inverse dynamics algorithm that computes the forces at the actuated joints corresponding to a prescribed motion. First, the inverse dynamics of the reduced system are computed according to an open-chain formulation, and second, the actuator forces are computed for the closed-chain manipulator using a physical interpretation of Lagrange multipliers. Kleinfinger and Khalil [1986] also developed a comparable computational scheme. Murray and Lovell [1989] made the comment that although these methods are general and systematic, there are many possible representations and choices for the constraint condition which will become very complicated when extended to multi-loop closed link mechanisms.

Based on the same idea of a virtual cut, Nakamura and Ghodoussi [1988] derived the torques applied at the active joints by projecting the generalised force vector of the unconstrained tree structure system. This was accomplished with a linear map incorporating the Jacobian of the passive joints with respect to the active joints. The procedure eliminated the necessity of calculating Lagrange multilpliers and hence reduced the computational burden. García de Jalón and Bayo [1994] suggested another way to deal with constraints which is to insert the constraints on to the Lagrange equations by means of a penalty formulation.

The approach used here is similar to that of Nakamura and Ghodoussi [1988] and is also covered in García de Jalón and Bayo [1994] and Shabana [1994]. Since the designer may wish to know the reaction forces exerted by the links, the Lagrange multilpliers are calculated, although they need not be if only the actuated forces are required.

The motivation behind analysing the dynamics of these mechanisms is to provide a dynamic comparison between similar mechanisms and to eventually assist with the design of the real manipulator. The method of dynamic analysis used here lends itself to very quick development times, unlike the method of static analysis in Chapter 6 where any change in configuration requires much effort. Thus, not only were the two and three dof mechanisms analysed, but also similar derivatives, which may in fact prove to be more appropriate for beam aiming applications. The dynamic analysis produced a full set of torques and forces experienced by each link so that the designer could use this information to obtain an optimum design.

In the following sections the dynamics of the mechanism itself are derived. Although five mechanisms were analysed, only the dynamics for the two dof prismatically actuated mechanism shown in figure 9.1 will be outlined in detail. The notation is general enough so that the extrapolation and solution of the dynamics of other parallel manipulators is straight forward.

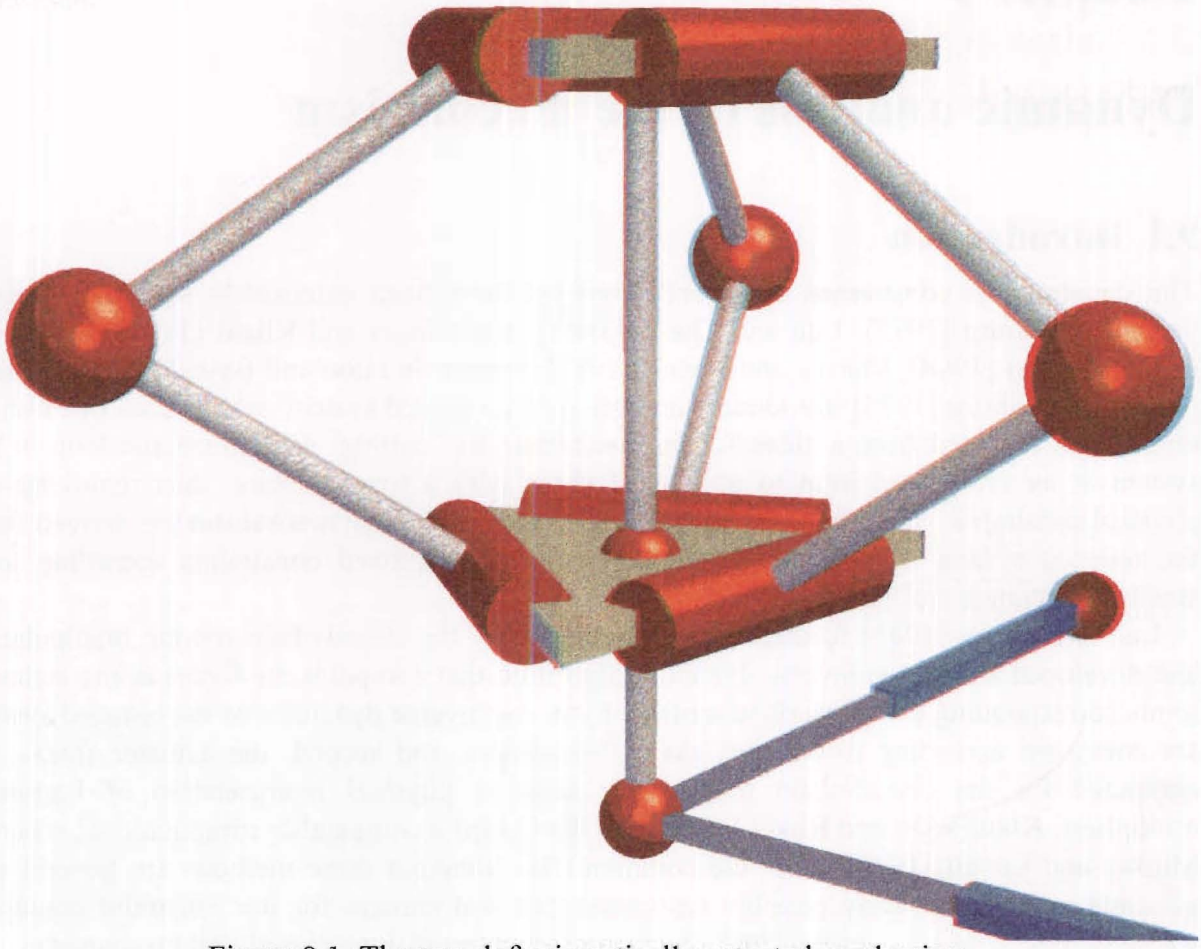


Figure 9.1 The two dof prismatically actuated mechanism.

9.2 Position kinematics

The position kinematics has for the most part been solved in Chapter 3. Although only the kinematics for the two and three dof rotary actuated mechanisms was developed, the kinematics for the other mechanisms are very similar. The calculation of the positions of the passive joints was also not covered, but is a fairly simple exercise using the dot product rule and comparing components of rotation matrices to extract the Euler angles of the multi degree of freedom joints. Please note that the coordinate frames, link and joint numbering system for this chapter differs from that of Chapter 3. The numbering system chosen (c.f. figure 9.2) reflects the link to link relative nature of the notation used in the dynamic formulations. A detailed description of the coordinate systems used to model the two dof prismatically actuated mechanism is given in Appendix E.

Once the motion of the satellite and the ship have been specified; the position of the mechanism at any instant in time can be calculated using similar inverse kinematic relationships to those derived in Chapter 5.

9.3 Velocity kinematics

The velocity kinematics involves calculating the velocities of all links in the mechanism given some prescribed satellite motion and ship motion. This problem is approached differently for parallel mechanisms as not all the joints are actuated as with serial robots.

9.3.1 Calculation of the joint rates $\dot{\beta}$

Consider the model of the two dof prismatically actuated mechanism shown below in figure 9.2.

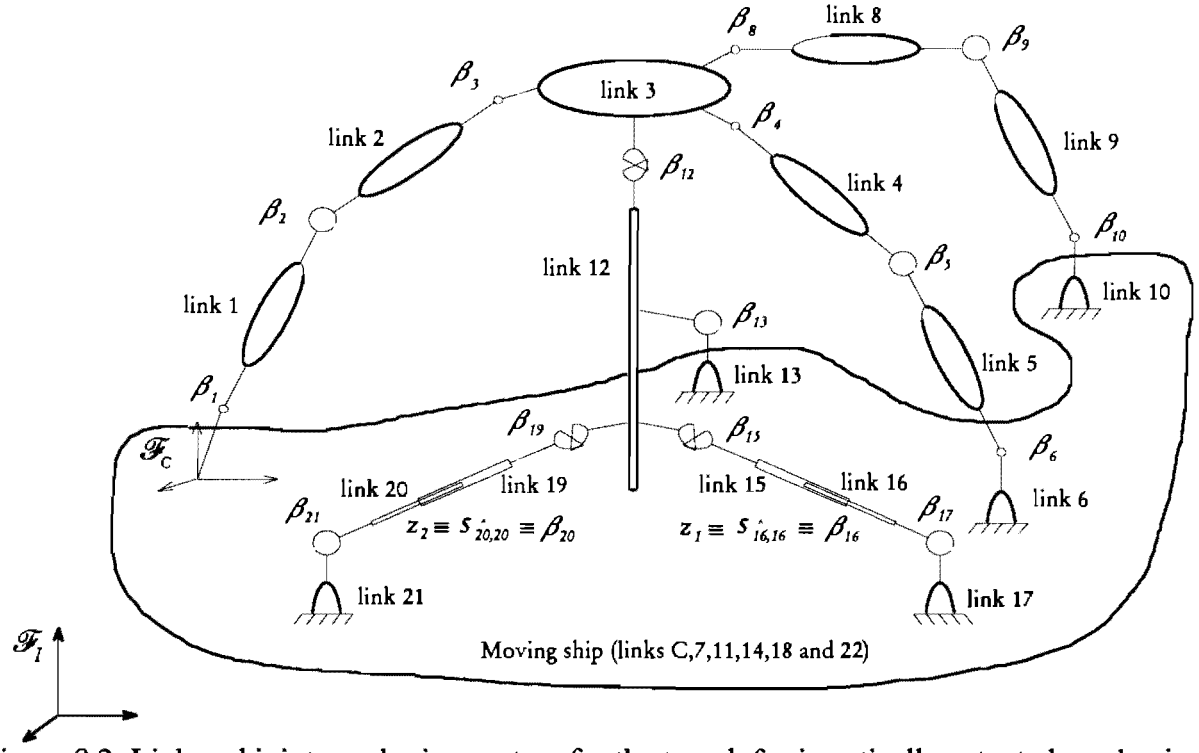


Figure 9.2 Link and joint numbering system for the two dof prismatically actuated mechanism.

From figure 9.2 it can be seen that the velocities of the fictitious links l_6, l_{10}, l_{13}, l_{17}, l_{21} are zero relative to the ship, which although it is one link, for convenience is defined as links l_C, l_7, l_{11}, l_{14}, l_{18} and l_{22}. Also, since the frame $\mathcal{F}_C$ is fixed relative to the frames $\mathcal{F}_7$, $\mathcal{F}_{11}$, $\mathcal{F}_{14}$, $\mathcal{F}_{17}$, $\mathcal{F}_{22}$ it is possible to write

$$
\begin{aligned}
{}^{7}v_{6,C} &= {}^{7}\mathcal{J}_{6}\,{}^{6}v_{6,7} &&= \mathbf{0},\\
{}^{11}v_{10,C} &= {}^{11}\mathcal{J}_{10}\,{}^{10}v_{10,11} &&= \mathbf{0},\\
{}^{14}v_{13,C} &= {}^{14}\mathcal{J}_{13}\,{}^{13}v_{13,14} &&= \mathbf{0},\\
{}^{18}v_{17,C} &= {}^{18}\mathcal{J}_{17}\,{}^{17}v_{17,18} &&= \mathbf{0},\\
{}^{22}v_{21,C} &= {}^{22}\mathcal{J}_{21}\,{}^{21}v_{21,22} &&= \mathbf{0},
\end{aligned}
$$

where the interbody transformation matrices

$$
{}^{7}\mathcal{J}_{6} = {}^{11}\mathcal{J}_{10} = {}^{14}\mathcal{J}_{13} = {}^{18}\mathcal{J}_{17} = {}^{22}\mathcal{J}_{21} = \mathbf{1}
$$

since the links l_6, l_{10}, l_{13}, l_{17}, l_{21} are fixed to the ship and their body fixed frames are aligned with frames $\mathcal{F}_7$, $\mathcal{F}_{11}$, $\mathcal{F}_{14}$, $\mathcal{F}_{17}$, and $\mathcal{F}_{22}$ respectively.

Applying these constraints to the mechanism yields

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \\ \\ v_{mc} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{9.1}$$

where v_{mc} is the global generalised velocity of the mechanism with respect to the moving ship's frame $\mathcal{F}_C$. For the two dof prismatically actuated mechanism in particular,

$$v_{mc} = ({}^{1}v_{1,C}, {}^{2}v_{2,C}, {}^{3}v_{3,C}, {}^{4}v_{4,C}, {}^{5}v_{5,C}, {}^{6}v_{6,C}, {}^{8}v_{8,C}, {}^{9}v_{9,C}, {}^{10}v_{10,C}, {}^{12}v_{12,C}, {}^{13}v_{13,C}, {}^{15}v_{15,C}, {}^{16}v_{16,C}, {}^{17}v_{17,C}, {}^{19}v_{19,C}, {}^{20}v_{20,C}, {}^{21}v_{21,C})^{\mathrm{T}}.$$

Equation (9.1) can be rewritten as

$$\mathbf{X}_{mc}\, v_{mc} = \mathbf{0}. \tag{9.2}$$

Since the velocity $v_{mc} = \mathcal{T}_{mc}\, \mathcal{G}_{mc}\, \mathbf{P}_{mc} \dot{\beta}_{mc}$ (c.f. section 8.5.2) equation (9.2) can be written as

$$\mathbf{X}_{mc}\, \mathcal{T}_{mc}\, \mathcal{G}_{mc}\, \mathbf{P}_{mc} \dot{\beta}_{mc} = \mathbf{0}$$

$$\text{or } \mathbf{Y}_{mc}\, \dot{\beta}_{mc} = \mathbf{0} \tag{9.3}$$

where

$$\mathbf{Y}_{mc} = \mathbf{X}_{mc}\, \mathcal{T}_{mc}\, \mathcal{G}_{mc}\, \mathbf{P}_{mc}$$

and $\dot{\beta}_{mc}$ is the rate vector containing both the independent $\dot{z}$ and dependent $\dot{\beta}_d$ rates of the mechanism only. For the two dof prismatically actuated mechanism in particular,

$$\dot{\beta}_{mc} = (\dot{\beta}_1, \dot{\beta}_2, \dot{\beta}_3, \dot{\beta}_4, \dot{\beta}_5, \dot{\beta}_6, \dot{\beta}_8, \dot{\beta}_9, \dot{\beta}_{10}, \dot{\beta}_{12}, \dot{\beta}_{13}, \dot{\beta}_{15}, \dot{\beta}_{16}, \dot{\beta}_{17}, \dot{\beta}_{19}, \dot{\beta}_{20}, \dot{\beta}_{21})^{\mathrm{T}}$$

and a detailed description of the matrices $\mathcal{T}_{mc}$, $\mathcal{G}_{mc}$ and $\mathbf{P}_{mc}$ can be found in Appendix E.

Discussion of equation (9.3)

Equation (9.3) indicates that the rate vector $\dot{\beta}_{mc}$ of the multibody system, at a specific position, belongs to the null space* of the Jacobian matrix $\mathbf{Y}_{mc}$ of the constraint equations. The theory of linear systems of equations (c.f. Strang [1976]) establishes that if the matrix $\mathbf{Y}_{mc}$ has m independent rows and n columns then the null space of $\mathbf{Y}_{mc}$ is the sub space of the possible or allowable motions (rates), in the sense that any possible rate vector compatible with the constraint equations must belong to this sub space. The dimension of the space of the allowable motions is the number of independent degrees of freedom f of the multibody system (for the mechanism shown in figure 9.2, f=2). For a more comprehensive overview of the problem see Chapter 3 of García de Jalón and Bayo [1994].

Given that the vector $\dot{\beta}_{mc}$ characterises the rates of the multibody system with n dependent coordinates, then the rates of the multibody system with a lower number of variables are obtained by constructing a basis for the null space. Thus the rates of the system can be represented by means of new vector $\dot{z}$, whose components are those of the vector $\dot{\beta}_{mc}$ on the chosen null space basis. If $\mathbf{r}_i$ is a set of f linearly independent vectors that constitute a basis of the null space of $\mathbf{Y}_{mc}$, then any dependent rate vector $\dot{\beta}_{mc}$ can be expressed as a linear combination of this basis as follows

$$\dot{\beta}_{mc} = \mathbf{r}_1 \dot{z}_1 + \mathbf{r}_2 \dot{z}_2 + \ldots + \mathbf{r}_f \dot{z}_f \qquad \text{f - no. of independent dof}$$

* null space - set of vectors in $\dot{\beta}_{mc}$ that $\mathbf{Y}_{mc}$ maps into $\mathbf{0}$.

where $\dot{z}_1 \ldots \dot{z}_f$ are the coefficients of the linear combination, namely the independent rates of the mechanism.

Now let $\mathbf{R}_{mc}$ be the matrix whose columns are the vectors $\mathbf{r}_i$ so that the above expression can be written in matrix notation as

$$\dot{\beta}_{mc} = \mathbf{R}_{mc}\ \dot{z}.$$

Once $\mathbf{R}_{mc}$ is found there is a relationship between the rates pertaining to the passive joints $\dot{\beta}_d$ (dependent dof) and the rates pertaining to the actuated joints $\dot{z}$ (independent dof).

9.3.2 Determination of the projection matrix R

There are a number of different methods which are used to obtain the matrix $\mathbf{R}_{mc}$ from $\mathbf{Y}_{mc}$. A method based on the Singular Value decomposition of $\mathbf{Y}_{mc}$ is used here. Alternative methods are described by García de Jalón and Bayo [1994], Shabana [1994], and involve the use of either QR decomposition, Gaussian triangularization or Orthogonalization methods.

Singular Value decomposition (SV) is a generalisation of the eigenvalue and eigenvector concept applicable to rectangular matrices. The SV decomposes a rectangular matrix such as $\mathbf{Y}_{mc}$ into the following

$$\mathbf{Y}_{mc} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$$

or written more fully as

$$\underset{m\times n}{\begin{bmatrix} \\ \mathbf{Y}_{mc} \\ \\ \end{bmatrix}} = \underset{m\times m}{\begin{bmatrix} \\ \mathbf{U} \\ \\ \end{bmatrix}} \underset{m\times f}{\begin{bmatrix} \ddots & \vdots & \\ \mathbf{S}_d & \mathbf{O} & \\ & \vdots & \\ & \mathbf{O} & \ddots \end{bmatrix}} \underset{f\times n}{\begin{bmatrix} & \mathbf{V}_d^T & \\ & & \\ \cdots & \cdots & \cdots \\ & \mathbf{V}_i^T & \end{bmatrix}}$$

For the two dof prismatically actuated mechanism, the total number of freedoms in the mechanism n=32 and the number of dependent freedoms in the mechanism m=30. From mobility theory it can be shown that f=n-m with the exception of mechanisms with special mobilities explained by Phillips [1984] and those having superfluous freedoms, e.g. a twist about the strut of the two dof rotary actuated mechanism shown in figure 1.9.

Matrix **U** is orthogonal and it is of size m×m. The matrix **S** can be decomposed into two sub matrices $\mathbf{S}_d$ which is a diagonal matrix of size m×m, that contains the singular values along its diagonal, and a zero matrix given by f=m-n columns. Matrix $\mathbf{V}^T$ is orthogonal and of size n×n. It can be decomposed into two sub matrices $\mathbf{V}_d^T$ and $\mathbf{V}_i^T$ of sizes m×n and f×n respectively, according to the partition in **S**. The most important property of the SV decomposition concerning the problem at hand is that the rows of the matrix $\mathbf{V}_i^T$ constitute an orthogonal basis of the null space of the matrix $\mathbf{Y}_{mc}$. The proof of this result is outlined as follows: since the columns of the matrices $\mathbf{V}_d$ and $\mathbf{V}_i$ are orthogonal vectors then

$$\mathbf{V}_d^T\mathbf{V}_i = \mathbf{O}$$

and since it is clear that

$$\mathbf{Y}_{mc} = \mathbf{U}\,\mathbf{S}_d\,\mathbf{V}_d^T$$

then these two equations imply that

$$\mathbf{Y}_{mc}\,\mathbf{V}_i = \mathbf{O}$$

$$\text{or } \mathbf{Y}_{mc}\,\mathbf{R}_{mc} = \mathbf{O} \qquad \text{where } \mathbf{R}_{mc} = \mathbf{V}_i. \tag{9.4}$$

It is necessary to normalise $\mathbf{R}_{mc}$, since although a basis for the null space of $\mathbf{Y}_{mc}$ has been found, the correct scaling has to be found. This can be done by finding an invertible matrix ψ so that the independent rates $\dot{z}$ map directly to themselves into the rate vector $\dot{\beta}_{mc}$. More specifically in the case of the two dof prismatically actuated mechanism the two independent rates $\dot{z}_1$ and $\dot{z}_2$ should be directly mapped into the 23rd and 29th entries of the vector $\dot{\beta}_{mc}$ which correspond to the specific joint rates $\dot{\beta}_{16}$ and $\dot{\beta}_{20}$ (c.f. figure 9.2). Thus a matrix ψ needs to be found so that the following is true

$$\begin{bmatrix} \mathrm{R}_{mc(23,1)} & \mathrm{R}_{mc(23,2)} \\ \mathrm{R}_{mc(29,1)} & \mathrm{R}_{mc(29,2)} \end{bmatrix} \psi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_a\,\psi = 1$$

$$\Rightarrow \psi = \mathbf{R}_a^{-1}.$$

The normalised matrix $\hat{\mathbf{R}}_{mc}$ is then given by

$$\hat{\mathbf{R}}_{mc} = \mathbf{R}_{mc}\,\psi \tag{9.5}$$

and the rate vector normalised with respect to the independent rates is now given by

$$\dot{\beta}_{mc} = \hat{\mathbf{R}}_{mc}\,\dot{z}\,. \tag{9.6}$$

Equation (9.6) provides a relationship between the rates pertaining to the actuated joints $\dot{z}$ (independent dof) and the mechanism rate vector $\dot{\beta}_{mc}$, which contains both the passive rates $\dot{\beta}_d$ (dependent dof) and the actuated joint rates $\dot{z}$.

9.3.3 Calculation of the independent (actuated) joint rates

This problem can be solved in one of two ways which are outlined below.

1) A Jacobian matrix can be formulated symbolically relating the platform pointing rates $\dot{\theta}_2$ and $\dot{\theta}_{3a}$ to the independent (actuated) joint rates $\dot{z}$. This symbolic approach to formulating the Jacobian matrix was developed for the three dof rotary actuated mechanism only, as it soon became obvious that it was too time consuming to do the same for the other mechanisms. For this reason a second approach described in the next section was used. The symbolic formulation of the Jacobian matrix for the three dof mechanism is included in Appendix C. If one compares the complexity of both approaches it is obvious that the second approach is vastly superior.

2) Given the pointing rates $\dot{\theta}_2$ and $\dot{\theta}_{3a}$ describing the motion of the platform relative to the moving ship's frame $\mathcal{F}_C$, the angular velocity of the platform ${}^3\omega_{3,C}$ can be calculated (see Appendix D) if the mechanism is assumed to be symmetrical through the spherical joints. Knowing the angular velocity of the platform then it is possible to calculate the independent (actuated) joint rates $\dot{\mathbf{z}}$ as follows: from section 8.5.2 the generalised velocity of the platform (link 3) relative to the ship $\mathcal{F}_C$ can be written as

$$
\begin{aligned}
{}^3\upsilon_{3,C} &= \begin{bmatrix} {}^3\mathcal{T}_1 & {}^3\mathcal{T}_2 & 1 \end{bmatrix} \begin{bmatrix} {}^1\mathcal{G}_{1,1} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & {}^2\mathcal{G}_{2,2} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & {}^3\mathcal{G}_{3,3} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{P}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{P}_3 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{R}}_{(1,1)} & \hat{\mathbf{R}}_{(1,2)} \\ \hat{\mathbf{R}}_{(2,1)} & \hat{\mathbf{R}}_{(2,2)} \\ \hat{\mathbf{R}}_{(3,1)} & \hat{\mathbf{R}}_{(3,2)} \\ \hat{\mathbf{R}}_{(4,1)} & \hat{\mathbf{R}}_{(4,2)} \\ \hat{\mathbf{R}}_{(5,1)} & \hat{\mathbf{R}}_{(5,2)} \end{bmatrix} \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} \\
&= \mathcal{T}_{b_1} \mathcal{G}_{b_1} \mathbf{P}_{b_1} \hat{\mathbf{R}}_{b_1} \dot{\mathbf{z}}
\end{aligned}
$$

The above can be written as

$$
\begin{pmatrix} {}^3\mathbf{v}_{3,C} \\ {}^3\omega_{3,C} \end{pmatrix} = \begin{bmatrix} \mathbf{A}_U \\ \mathbf{A}_L \end{bmatrix} \dot{\mathbf{z}}
$$

where $\mathbf{A}_U$ and $\mathbf{A}_L$ are the upper and lower 3×2 partitions respectively of the 6×2 matrix

$$
\mathbf{A} = \mathcal{T}_{b_1} \mathcal{G}_{b_1} \mathbf{P}_{b_1} \hat{\mathbf{R}}_{b_1} . \tag{9.7}
$$

The angular velocity of frame $\mathcal{F}_3$ attached to the platform (link 3) with respect to frame $\mathcal{F}_C$ is then given by

$$
{}^3\omega_{3,C} = \mathbf{A}_L \dot{\mathbf{z}}. \tag{9.8}
$$

The angular velocity of frame $\mathcal{F}_3$ with respect to frame $\mathcal{F}_C$ (c.f. Appendix D) can also be written as

$$
{}^3\omega_{3,C} = {}^3\mathbf{C}_{u_2} {}^{u_2}\mathbf{S}_0 \begin{pmatrix} -\dot{\theta}_{3a} \\ \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix} \tag{9.9}
$$

where $\dot{\theta}_{3a}$, $\dot{\theta}_2$ and $-\dot{\theta}_{3a}$ are a 3-2-3 Euler set of the known pointing rates of the platform. Combining equations (9.8) and (9.9) the pointing rates of the platform in terms of the actuated joint rates can be written as follows

$$
\begin{pmatrix} -\dot{\theta}_{3a} \\ \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix} = \mathbf{J}\dot{\mathbf{z}}
$$

where

$$
\mathbf{J} = \begin{bmatrix} \mathbf{J}_U \\ \mathbf{J}_L \end{bmatrix} = {}^{u_2}\mathbf{S}_0^{-1} {}^3\mathbf{C}_{u_2}^{-1} \mathbf{A}_L \tag{9.10}
$$

and where $\mathbf{J}_U$ and $\mathbf{J}_L$ are the upper 1×2 and lower 2×2 partitions respectively of the 3×2 matrix $\mathbf{J}$.

The generalised rate coordinates of the platform can now be written as a 3-2 Euler set in terms of the actuated joint rates as follows:

$$\begin{pmatrix} \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix} = \mathbf{J}_L \dot{z}. \tag{9.11}$$

The actuated joint rates can then be found in terms of the known platform pointing rates as follows:

$$\dot{z} = \mathbf{J}_L^{-1} \begin{pmatrix} \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix}.$$

9.4 Acceleration kinematics

The time derivative of the mechanism rate vector can be found as follows. The independent rates $\dot{z}$ can be defined as the projection of the rates $\dot{\beta}_{mc}$ on the rows of a constant (not time or position dependent) matrix $\mathbf{B}$

$$\dot{z} = \mathbf{B}\dot{\beta}_{mc}. \tag{9.12}$$

For the two dof prismatically actuated mechanism

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

where the ones are the 23rd and 29th entries respectively. Differentiating equation (9.3) with respect to time gives

$$\begin{aligned} \ddot{z} &= \dot{\mathbf{B}}\dot{\beta}_{mc} + \mathbf{B}\ddot{\beta}_{mc} \\ &= \mathbf{B}\ddot{\beta}_{mc}. \end{aligned} \tag{9.13}$$

Differentiating equation (9.3) yields

$$\begin{aligned} &\dot{\mathbf{Y}}_{mc}\dot{\beta}_{mc} + \mathbf{Y}_{mc}\ddot{\beta}_{mc} = \mathbf{0} \\ &\Rightarrow \mathbf{Y}_{mc}\ddot{\beta}_{mc} = -\dot{\mathbf{Y}}_{mc}\dot{\beta}_{mc}. \end{aligned} \tag{9.14}$$

The matrix $\dot{\mathbf{Y}}_{mc}$ is given by

$$\dot{\mathbf{Y}}_{mc} = \mathbf{X}_{mc}\dot{\mathcal{T}}_{mc}\mathcal{G}_{mc}\mathbf{P}_{mc} + \mathbf{X}_{mc}\mathcal{T}_{mc}\dot{\mathcal{G}}_{mc}\mathbf{P}_{mc} + \mathbf{X}_{mc}\mathcal{T}_{mc}\mathcal{G}_{mc}\dot{\mathbf{P}}_{mc}.$$

Note that $\dot{\mathbf{X}}_{mc} = \mathbf{0}$ since $\mathbf{X}_{mc}$ is constant. The matrices $\dot{\mathcal{T}}_{mc}$, $\dot{\mathcal{G}}_{mc}$, $\dot{\mathbf{P}}_{mc}$ are given in Appendix E. Equations (9.13) and (9.14) can be augmented to give

$$\begin{bmatrix} \mathbf{Y}_{mc} \\ \mathbf{B} \end{bmatrix} \ddot{\beta}_{mc} = \begin{bmatrix} -\dot{\mathbf{Y}}_{mc}\dot{\beta}_{mc} \\ \ddot{z} \end{bmatrix}. \tag{9.15}$$

Assuming that the matrix $\mathbf{B}$, in addition to being constant, also fulfills the condition of having f=n-m rows that are linearly independent from one another and are also linearly independent of the m rows of $\mathbf{Y}_{mc}$. With these assumptions the matrix in equation (9.15) can be inverted, and finding the vector $\ddot{\beta}_{mc}$ involves the solution of the following equation:

$$\ddot{\beta}_{mc} = \begin{bmatrix} \mathbf{Y}_{mc} \\ \mathbf{B} \end{bmatrix}^{-1} \begin{bmatrix} -\dot{\mathbf{Y}}_{mc}\hat{\mathbf{R}}_{mc}\dot{z} \\ \ddot{z} \end{bmatrix}. \tag{9.16}$$

9.4.1 Calculation of the time derivatives of independent joint rates

The time derivatives of independent joint rates can be found by differentiating equation (9.11) with respect to time to get

$$\begin{pmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_{3a} \end{pmatrix} = \dot{\mathbf{J}}_L \dot{\mathbf{z}} + \mathbf{J}_L \ddot{\mathbf{z}}.$$

Rearranging the above the time derivatives of the actuated joint rates can then be found in terms of the known platform pointing rates (and their time derivatives) as follows:

$$\ddot{\mathbf{z}} = \mathbf{J}_L^{-1} \left(\begin{pmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_{3a} \end{pmatrix} - \dot{\mathbf{J}}_L \dot{\mathbf{z}} \right).$$

The time derivative $\dot{\mathbf{J}}_L$ of the Jacobian matrix can be found by differentiating equation (9.10) with respect to time as follows

$$\dot{\mathbf{J}} = \begin{bmatrix} \dot{\mathbf{J}}_U \\ \dot{\mathbf{J}}_L \end{bmatrix} = {}^3\dot{\mathbf{S}}_0^{-1}\, {}^3\mathbf{C}_{u_2}^{-1} \mathbf{A}_L + {}^3\mathbf{S}_0^{-1}\, {}^3\mathbf{C}_{u_2}^{-1} \dot{\mathbf{A}}_L \qquad \text{since } {}^3\dot{\mathbf{C}}_{u_2}^{-1} = \mathbf{O}.$$

The matrix ${}^3\dot{\mathbf{S}}_0^{-1}$ can be found by using the product rule to get

$${}^3\dot{\mathbf{S}}_0^{-1} = -{}^3\mathbf{S}_0^{-1}\, {}^3\dot{\mathbf{S}}_0\, {}^3\mathbf{S}_0^{-1}.$$

The time derivative $\dot{\mathbf{A}}_L$ can be found by differentiating equation (9.7) with respect to time as follows

$$\dot{\mathbf{A}} = \dot{\mathcal{T}}_{b_l} \mathcal{G}_{b_l} \mathbf{P}_{b_l} \hat{\mathbf{R}}_{b_l} + \mathcal{T}_{b_l} \dot{\mathcal{G}}_{b_l} \mathbf{P}_{b_l} \hat{\mathbf{R}}_{b_l} + \mathcal{T}_{b_l} \mathcal{G}_{b_l} \dot{\mathbf{P}}_{b_l} \hat{\mathbf{R}}_{b_l} + \mathcal{T}_{b_l} \mathcal{G}_{b_l} \mathbf{P}_{b_l} \dot{\hat{\mathbf{R}}}_{b_l}.$$

9.4.2 Determination of the projection matrix $\dot{\hat{\mathbf{R}}}$

The matrix $\dot{\hat{\mathbf{R}}}_{b_l}$ can be found by rewriting equation (9.16) as

$$\begin{aligned} \ddot{\beta}_{mc} &= \begin{bmatrix} \mathbf{H} & \mathbf{K} \end{bmatrix} \begin{bmatrix} -\dot{\mathbf{Y}}_{mc} \hat{\mathbf{R}}_{mc} \dot{\mathbf{z}} \\ \ddot{\mathbf{z}} \end{bmatrix} \\ &= -\mathbf{H} \dot{\mathbf{Y}}_{mc} \hat{\mathbf{R}}_{mc} \dot{\mathbf{z}} + \mathbf{K} \ddot{\mathbf{z}}. \end{aligned}$$

Comparing the above with the time differentiation of equation (9.6)

$$\dot{\beta} = \dot{\hat{\mathbf{R}}} \dot{\mathbf{z}} + \hat{\mathbf{R}} \ddot{\mathbf{z}}$$

it can be seen that

$$\hat{\mathbf{R}}_{mc} = \mathbf{K}$$

and

$$\dot{\hat{\mathbf{R}}}_{mc} = -\mathbf{H} \dot{\mathbf{Y}}_{mc} \hat{\mathbf{R}}_{mc}.$$

The matrix $\dot{\hat{\mathbf{R}}}_{b_l}$ can be extracted from the relevant partition of $\dot{\hat{\mathbf{R}}}_{mc}$, for the two dof prismatically actuated mechanism $\dot{\hat{\mathbf{R}}}_{b_l} = \dot{\hat{\mathbf{R}}}_{mc}(1{:}5,1{:}2)$.

9.4.3 Global assembly of the rates and their time derivatives

Now that both the joint rates of the mechanism $\dot{\beta}_{mc}$ and the moving ship (c.f. section 7.2) are known (and their time derivatives), they can be assembled as combined global rate vectors for both the ship and mechanism as follows:

$$\dot{\beta} = \begin{pmatrix} \dot{s}_{\hat{A}A} \\ \dot{\psi} \\ \dot{\vartheta} \\ \dot{\beta}_{mc} \end{pmatrix} \quad \ddot{\beta} = \begin{pmatrix} \ddot{s}_{\hat{A}A} \\ \ddot{\psi} \\ \ddot{\vartheta} \\ \ddot{\beta}_{mc} \end{pmatrix}.$$

9.5 Dynamics

Recall from section 8.5.3 that the assembled system of the total generalised body forces for a serial multibody system can be written as

$$\mathcal{F}_T = \mathcal{T}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} + \mathcal{F}_{Ext}. \tag{9.17}$$

Equation (9.17) can be written with the unknown constraint forces λ explicitly included as follows

$$\mathcal{F}_T = \mathcal{T}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} - \mathbf{X}^T\lambda + \mathcal{F}_{Ext}. \tag{9.18}$$

Note that equation (9.18) includes not only the terms pertaining to the mechanism itself but also the terms pertaining to the ship and stabilised platform. A detailed description of the matrices $\mathcal{T}$, $\mathcal{G}$ and $\mathcal{F}_{Ext}$ can be found in Appendix E. With the inclusion of the dynamics of the moving ship and stabilised platform the matrix $\mathbf{X}$ is given by $\mathbf{X} = [\mathbf{O}\ \mathbf{X}_{mc}]$ or more specifically for the two dof prismatically actuated mechanism $\mathbf{X} = [\mathbf{O}_{30\times 18}\ \mathbf{X}_{mc}]$.

For the two dof prismatically actuated mechanism the unknown constraint forces λ are the forces exerted by links 6,10,13,17,21 on the ship, i.e.

$$\lambda = \begin{pmatrix} {}^{7}\mathcal{F}_{6,7} \\ {}^{11}\mathcal{F}_{10,11} \\ {}^{14}\mathcal{F}_{13,14} \\ {}^{18}\mathcal{F}_{17,18} \\ {}^{22}\mathcal{F}_{21,22} \end{pmatrix}.$$

9.5.1 Global constrained joint motion equations

The global motion equations are the same as in section 8.5.4 with the exception that the total generalised body force is now given by

$$\mathcal{F}_T = \mathcal{T}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} - \mathbf{X}^T\lambda + \mathcal{F}_{Ext}. \tag{9.19}$$

Recall equation (8.45) from section 8.5.4

$$\mathbf{M}\mathcal{T}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{F}_T - v^{\otimes}\mathbf{M}v - \mathbf{M}\left(\dot{\mathcal{T}}\mathcal{G}\mathbf{P} + \mathcal{T}\dot{\mathcal{G}}\mathbf{P} + \mathcal{T}\mathcal{G}\dot{\mathbf{P}}\right)\dot{\beta}. \tag{9.20}$$

Detailed descriptions of the matrices $v^{\otimes}\mathbf{M}v$ and $\mathbf{M}$ are given in Appendix E.

Substituting for (9.19) and $\mathcal{F}_{Non}$ (c.f. section 8.5.3) into (9.20) yields

$$\mathbf{M}\mathcal{T}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{T}^{-T}\mathcal{G}^{-T}\mathcal{F}_{Int} + \mathcal{F}_{Ext} + \mathcal{F}_{Non} - \mathbf{X}^T\lambda. \tag{9.21}$$

Pre multiplying (9.21) by $\mathbf{P}^T\mathcal{G}^T\mathcal{T}^T$ gives

$$\mathbf{P}^T\mathcal{G}^T\mathcal{T}^T\mathbf{M}\mathcal{T}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{P}^T\mathcal{F}_{Int} + \mathbf{P}^T\mathcal{G}^T\mathcal{T}^T\mathcal{F}_{Ext} + \mathbf{P}^T\mathcal{G}^T\mathcal{T}^T\mathcal{F}_{Non} - \mathbf{P}^T\mathcal{G}^T\mathcal{T}^T\mathbf{X}^T\lambda. \tag{9.22}$$

Substitution of equation (8.43) into (9.22) permits the elimination of the constraint forces to get the global constrained joint motion equation

$$\mathbf{M}_{\beta\beta}\ddot{\beta} = \mathbf{1}_p\mathcal{F}_C + \mathcal{F}_E + \mathcal{F}_{Non,\beta} - \mathbf{Y}^T\lambda. \tag{9.23}$$

Detailed descriptions of the matrices $\mathbf{M}_{\beta\beta}$, $\mathcal{F}_E$ and $\mathcal{F}_{Non,\beta}$ are given in section 8.5.4. The matrix $\mathbf{Y}$ which includes the terms pertaining to the ship and stabilised platform is given by $\mathbf{Y} = [\mathbf{O}\ \mathbf{Y}_{mc}]$ or more specifically for the two dof prismatically actuated mechanism $\mathbf{Y} = [\mathbf{O}_{30\times 9}\ \mathbf{Y}_{mc}]$.

The closed loop actuator (control) forces expressed in z space (the independent coordinate space) F_z can be calculated by eliminating the unknown constraint forces λ from equation (9.23) by premultiplying by $\hat{\mathbf{R}}^T$ as follows:

$$\begin{aligned}\hat{\mathbf{R}}^T\mathbf{M}_{\beta\beta}\ddot{\beta} &= \hat{\mathbf{R}}^T\mathbf{1}_p\mathcal{F}_C + \hat{\mathbf{R}}^T\mathcal{F}_E + \hat{\mathbf{R}}^T\mathcal{F}_{Non,\beta} - \hat{\mathbf{R}}^T\mathbf{Y}^T\lambda \\ &= \hat{\mathbf{R}}^T\mathbf{1}_p\mathcal{F}_C + \hat{\mathbf{R}}^T\mathcal{F}_E + \hat{\mathbf{R}}^T\mathcal{F}_{Non,\beta} - (\mathbf{Y}\hat{\mathbf{R}})^T\lambda \\ &= \hat{\mathbf{R}}^T\mathbf{1}_p\mathcal{F}_C + \hat{\mathbf{R}}^T\mathcal{F}_E + \hat{\mathbf{R}}^T\mathcal{F}_{Non,\beta} \qquad \text{since } \mathbf{Y}\hat{\mathbf{R}} = \mathbf{O}.\end{aligned}$$

Note that the matrix $\hat{\mathbf{R}}$ includes the terms pertaining to the ship and stabilised platform and is given by $\hat{\mathbf{R}} = \begin{bmatrix}\mathbf{O}\\ \hat{\mathbf{R}}_{mc}\end{bmatrix}$. More specifically for the two dof prismatically actuated mechanism $\hat{\mathbf{R}} = \begin{bmatrix}\mathbf{O}_{9\times 2}\\ \hat{\mathbf{R}}_{mc}\end{bmatrix}$.

The closed loop actuator (control) forces F_z are then given by

$$\begin{aligned}F_z &= \hat{\mathbf{R}}^T\mathbf{1}_p\mathcal{F}_C = \hat{\mathbf{R}}^T\mathbf{M}_{\beta\beta}\ddot{\beta} - \hat{\mathbf{R}}^T\mathcal{F}_E - \hat{\mathbf{R}}^T\mathcal{F}_{Non,\beta} \\ &= \hat{\mathbf{R}}^T(\mathbf{M}_{\beta\beta}\ddot{\beta} - \mathcal{F}_E - \mathcal{F}_{Non,\beta}).\end{aligned} \tag{9.24}$$

9.5.2 Calculation of the constraint forces λ

The constraint forces can be found by pre multiplying equation (9.23) by $\mathbf{Y}$ to get

$$\mathbf{M}_{\beta\beta}\ddot{\beta} = \mathbf{Y}\mathbf{1}_p\mathcal{F}_C - \mathbf{Y}\mathcal{F}_E - \mathbf{Y}\mathcal{F}_{Non,\beta} - \mathbf{Y}\mathbf{Y}^T\lambda.$$

The constraint forces are then given by

$$\lambda = -(\mathbf{Y}\mathbf{Y}^T)^{-1}\mathbf{Y}(\mathbf{M}_{\beta\beta}\ddot{\beta} - \mathbf{1}_p\mathcal{F}_C - \mathcal{F}_E - \mathcal{F}_{Non,\beta})$$

where the control forces in β space are given by

$$\boldsymbol{F}_c = \begin{pmatrix} \mathbf{0}_{9\times 1} \\ 0 \\ 0 \\ \vdots \\ F_{z(1)} \\ 0 \\ \vdots \\ F_{z(2)} \end{pmatrix}.$$

Note that for the two dof prismatically actuated mechanism, the control forces $F_{z(1)}$ and $F_{z(2)}$ would be the 32nd and 38th entries respectively. Once the constraint forces λ have been calculated the remaining joint reactions can be calculated by using either a recursive or global approach. Using the global approach equation (9.21) can be premultiplied by $\mathcal{G}^T\mathcal{J}^T$ to get

$$\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathcal{F}_{Int} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non} - \mathcal{G}^T\mathcal{J}^T\mathbf{X}^T\lambda. \qquad (9.25)$$

Substitution of equation (8.43) leads to

$$\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{P}\mathcal{F}_C + \mathbf{Q}\mathcal{F}_{Con} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non} - \mathcal{G}^T\mathcal{J}^T\mathbf{X}^T\lambda. \qquad (9.26)$$

Next premultiplying (9.26) by $\mathbf{Q}^T$ and eliminating the control torques $\mathcal{F}_C$ using the projection matrix property $\mathbf{Q}^T\mathbf{P} = \mathbf{O}$ gives

$$\mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} = \mathbf{Q}^T\mathbf{Q}\mathcal{F}_{Con} + \mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Ext} + \mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathcal{F}_{Non} - \mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T\mathbf{X}^T\lambda.$$

The global constraint forces are then given by

$$\mathcal{F}_{Con} = \mathbf{1}_q\mathbf{Q}^T\mathcal{G}^T\mathcal{J}^T(\mathbf{M}\mathcal{J}\mathcal{G}\mathbf{P}\ddot{\beta} - \mathcal{F}_{Ext} - \mathcal{F}_{Non} + \mathbf{X}^T\lambda).$$

9.6 Summary of procedures to calculate the inverse dynamics

Below is a summary of the steps carried out when calculating the inverse dynamics of the mechanism.

Step 1	Set path parameters of the tracked object (c.f. section 7.1)
Step 2	Calculate pointing angles ϕ and time derivatives $\dot{\phi}, \ddot{\phi}$ of tracked object w.r.t frame $\mathcal{F}_I$ (c.f. section 7.1)
Step 3	Set ship and stabilised platform motion parameters (c.f. section 7.2)
Step 4	Calculate ${}^A s_{\hat{A}A}, \psi, \vartheta$ and time derivatives (c.f. section 7.2)
Step 5	Calculate pointing angles θ and time derivatives $\dot{\theta}, \ddot{\theta}$ of tracked object w.r.t frame $\mathcal{F}_C$ (c.f. section 7.3)
Step 6	Calculate geometric parameters of mechanism (c.f. Chapter 5)
Step 7	Formulate velocity constraint matrix $\mathbf{X}_{mc}$ (c.f. equation 9.1)
Step 8	Formulate matrices $\mathcal{J}_{mc}$, $\mathcal{G}_{mc}$ and $\mathbf{P}_{mc}$ (c.f. Appendix E)
Step 9	Calculate rate constraint matrix $\mathbf{Y}_{mc}$ (c.f. equation 9.3)
Step 10	Calculate rate projection matrix $\hat{\mathbf{R}}_{mc}$ (c.f. section 9.3.2)
Step 11	Calculate independent (actuated) joint rates $\dot{z}$ (c.f. section 9.3.3)

Step 12 Formulate matrices $\dot{\mathfrak{J}}_{mc}$, $\dot{\mathcal{G}}_{mc}$ and $\dot{\mathbf{P}}_{mc}$ (c.f. Appendix E)

Step 13 Calculate matrix $\dot{\mathbf{Y}}_{mc}$ (c.f. section 9.4)

Step 14 Calculate rate projection matrix $\dot{\hat{\mathbf{R}}}_{mc}$ (c.f. section 9.4.2)

Step 15 Calculate time derivatives of independent (actuated) joint rates $\dot{z}$ (c.f. section 9.4.1)

Step 16 Assemble combined global rate vectors for both the ship and mechanism $\dot{\beta}$, $\ddot{\beta}$ (c.f. section 9.4.3)

Step 17 Calculate the assembled system of generalised velocities v for the entire system

Step 18 Calculate the column matrix $v^{\otimes}\mathbf{M}v$

Step 19 Calculate the non linear force column matrix $\mathcal{F}_{Non,\beta}$ (c.f. section 8.5.4)

Step 20 Calculate the external force column matrix $\mathcal{F}_E$ (c.f. section 8.5.4)

Step 21 Calculate the mass matrix $\mathbf{M}_{\beta\beta}$ (c.f. section 8.5.4)

Step 22 Calculate the control (actuator) forces F_z (c.f. equation 9.23)

Step 23 Calculate the constraint forces λ (c.f. section 9.5.2)

Step 24 Calculate the constraint forces $\mathcal{F}_{Con}$ (c.f. section 9.5.2)

Step 25 Check whether the simulation time is exceeded

9.7 Error checking of the velocity/acceleration kinematics and the dynamic model

Short of taking a completely different approach to the problem, the model developed in this chapter cannot be completely validated but most of the model can be verified by simple checks. All rates (and their time derivatives), velocities and accelerations can be checked by small perturbations, e.g. a rate can be checked by taking the difference between the positions at two close points in time and dividing by the change in time. The time derivatives of the interbody transformation matrices and the projection matrices can be checked in the same way: thus the velocity and acceleration kinematics can be completely validated.

Assuming that the kinematics are correct, the static model developed in Chapter 6 can be used to verify the zeroth and first moments of mass entries in the mass matrices **M**. The static portion of the dynamic analyses developed in this chapter can be isolated by ignoring the inertial effects: hence solving for the actuator forces using the external forces (forces due to gravity) only, equation (9.24) becomes

$$F_z = \hat{\mathbf{R}}^T\left(-\mathcal{F}_E\right).$$

Figures 9.3 and 9.4 show the actuator requirements for both the static analysis developed in Chapter 6 and the static portion of the analysis developed in this chapter. Both plots are for 360° horizon sweep of the mechanism only without an antenna load. The plots show very good correlation between both analysis, which was anticipated since both physical models are identical. The second moments of mass (moments of inertia) cannot be verified explicitly, but an energy comparison between the energy in the moving mechanism with the work done by the actuators could be used to solve this problem.

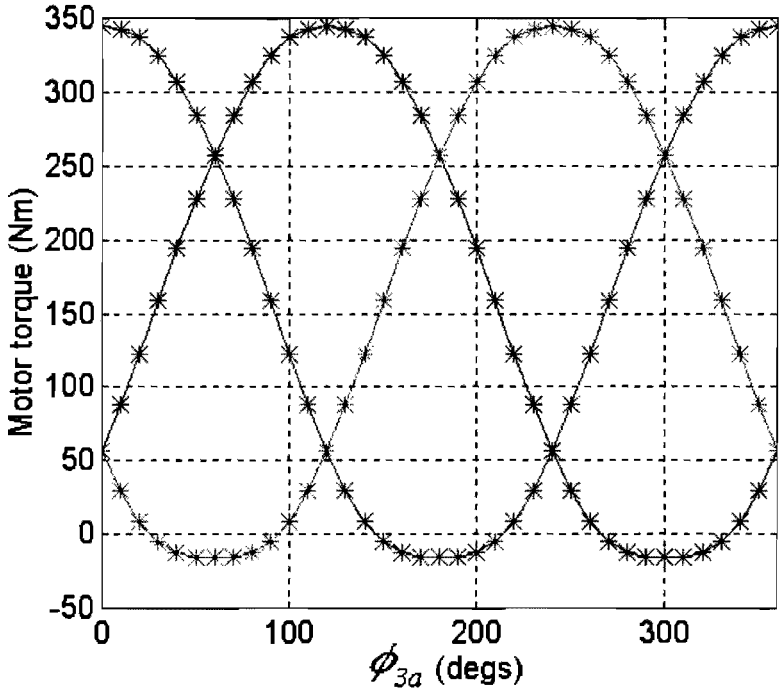


Figure 9.3 Comparison of the static analyses developed in Chapters 6 and 9 for the three dof rotary actuated mechanism.

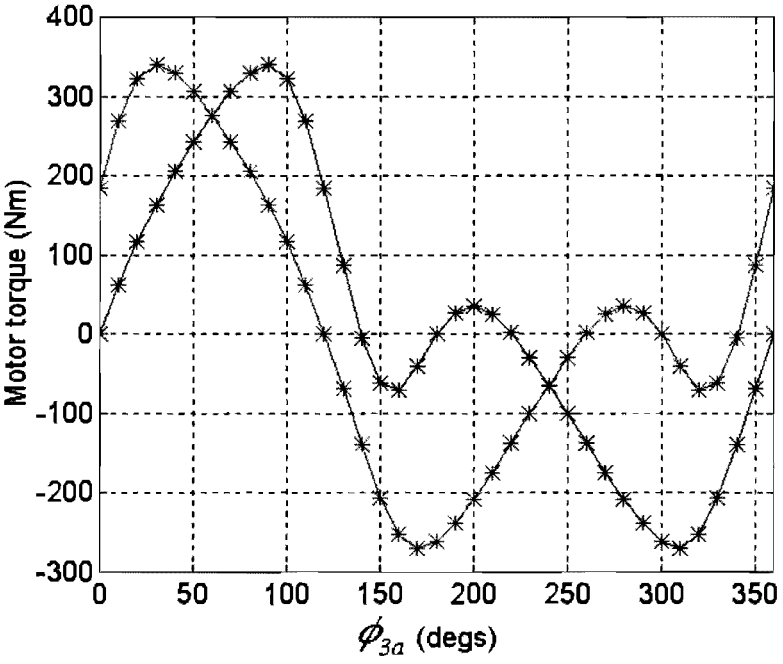


Figure 9.4 Comparison of the static analyses developed in Chapters 6 and 9 for the two dof rotary actuated mechanism.

Chapter 10

Results and observations

10.1 Introduction

In this chapter the simulated results of five mechanisms are compared. The five mechanisms are formed from the two and three dof parallel mechanisms and incorporate different actuator arrangements and counter balancing schemes. Counter balancing of the links is introduced to lower the actuator demands to acceptable levels. The problems encountered by the mechanisms at certain low elevation angles are discussed. A maritime simulation is also carried out to show the effect of mounting the mechanism on a moving ship.

10.2 Servo requirements

Rigid body dynamic models of the five mechanisms were developed based on the formulations developed in Chapters 7, 8 and 9. These models had the same geometric and mass properties as the robot built at the university with the inclusion of a 100 kg antenna. A 360° horizon sweep produced the worst case loading scenario; thus the servo requirements for the five mechanisms while moving in a slow (i.e. inertial effects ignored) 360° sweep around the horizon are plotted.

Servo requirements with counter balancing

Different combinations of counter balancing are then introduced and are shown to lower the servo requirements significantly. The options for counter balancing are that the lower arms may be counter balanced and/or the strut may be counter balanced in the case of the two dof mechanisms.

The many intermediate results obtained while manually altering the parameters are not presented here, and although no rigorous optimisation has been carried out, the results obtained from just simple manual parameter changes are encouraging. During the manual optimisation process, the variables altered were the angular offset of the counter balance from its particular arm and the offset distance and mass of the counter balance. It was noted that the best results were obtained when the angular offset was 180°. This may not be possible in practice owing to interference with the base. It may also be necessary to laterally offset the counter balances to avoid interference with the other counter balances and the extended strut. Thus, the exact results shown here may not be fully realisable since some reduction in the counter balance angular offset may be required. However the results obtained here can still be closely approached.

10.2.1 Servo requirements for the three dof rotary actuated mechanism

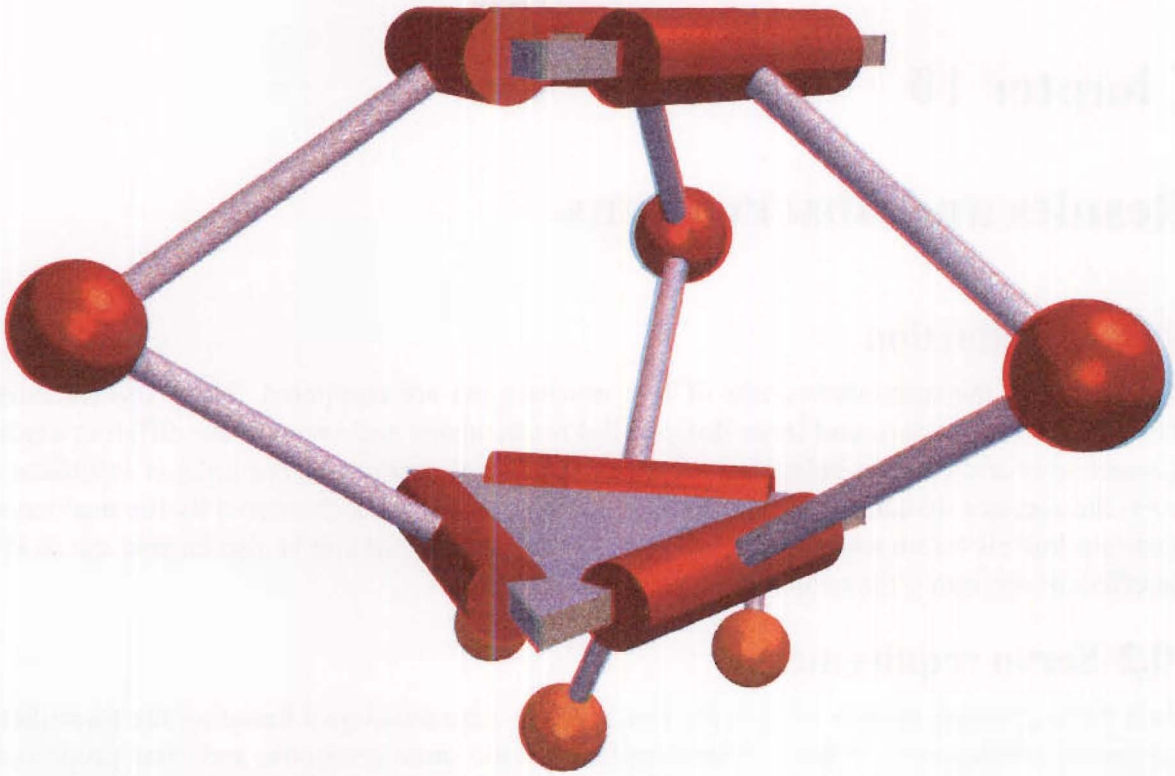


Figure 10.1 Three dof rotary actuated mechanism with counter balancing of the lower arms

The servo requirements for a horizon sweep are plotted below. Figure 10.3 clearly shows the significant improvements to be gained with counter balancing of the lower arms. Note that these plots were generated with a constant radial distance between the platform and base; thus some improvement could be gained if this redundant freedom was altered to maximise the stiffness (mechanical advantage) of the mechanism.

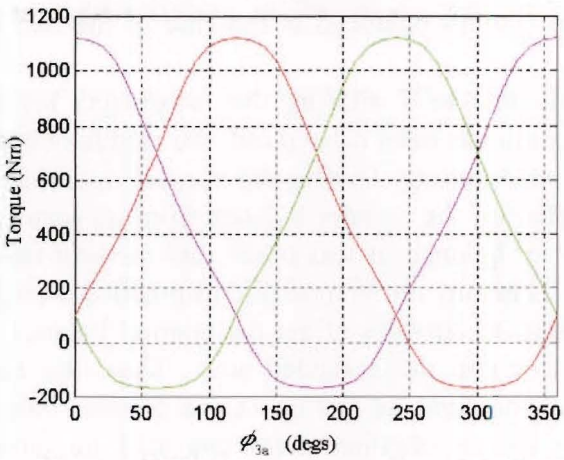


Figure 10.2 Actuator requirements for a horizon sweep of the three dof mechanism with no counter balancing.

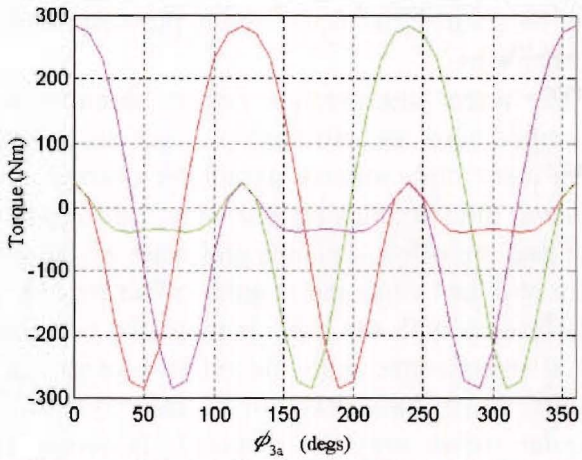


Figure 10.3 Actuator requirements for a horizon sweep of the three dof mechanism with counter balancing of the lower arms only.

10.2.2 Servo requirements for the two dof rotary actuated mechanism

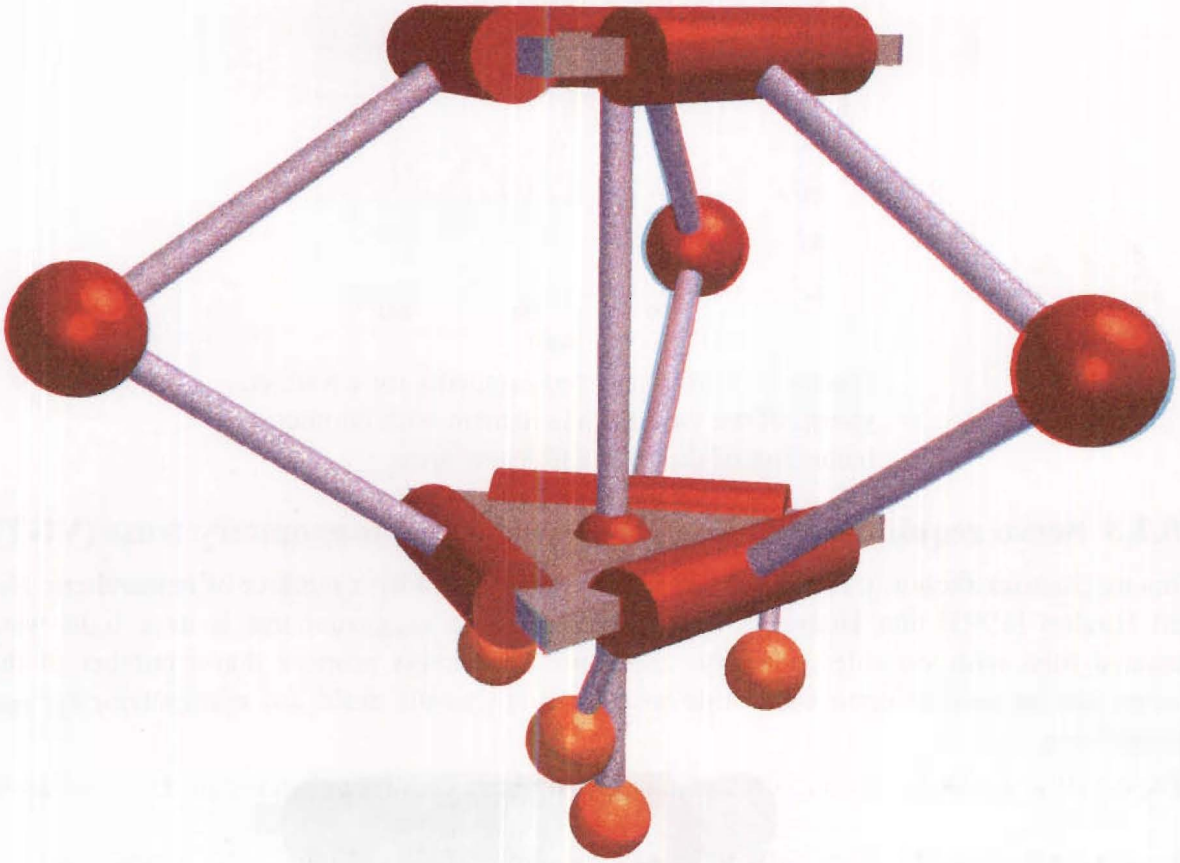


Figure 10.4 Two dof rotary actuated mechanism with counter balancing of the lower arms and strut.

The servo requirements for a horizon sweep of the two dof mechanism are plotted below. Figures 10.6 and 10.7 clearly show significant improvements to be gained with counter balancing of the strut and/or lower arms.

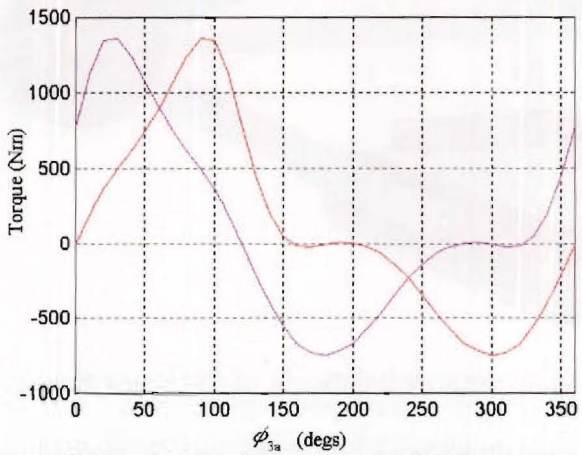


Figure 10.5 Actuator requirements for a horizon sweep of the two dof mechanism with no counter balancing.

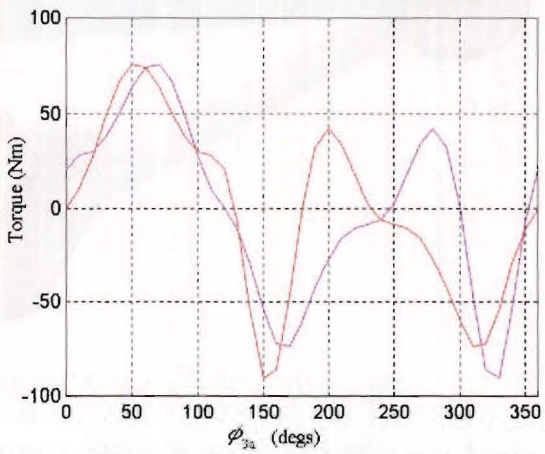


Figure 10.6 Actuator requirements for a horizon sweep of the two dof mechanism with counter balancing of the strut only.

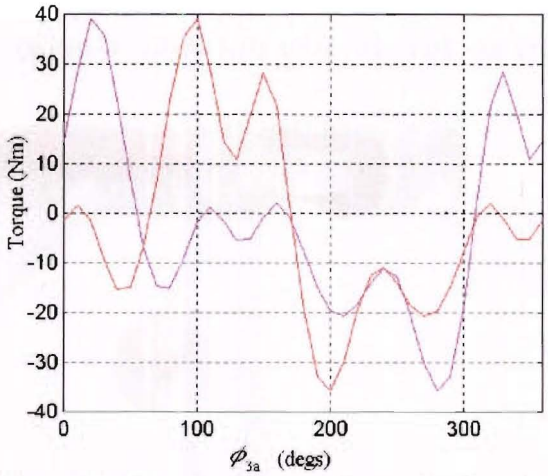


Figure 10.7 Actuator requirements for a horizon sweep of the two dof mechanism with counter balancing of the strut and lower arms.

10.2.3 Servo requirements for the three dof variable geometry truss (VGT)

This mechanism shown in figure 10.8 has been investigated by a number of researchers: Hertz and Hughes [1993] and Huang et al. [1996] where its suggested use is as a light weight actuated truss with variable geometry. The same researchers propose that a number of these trusses can be stacked upon each other to form a fairly stiff multi dof manipulator for space applications.

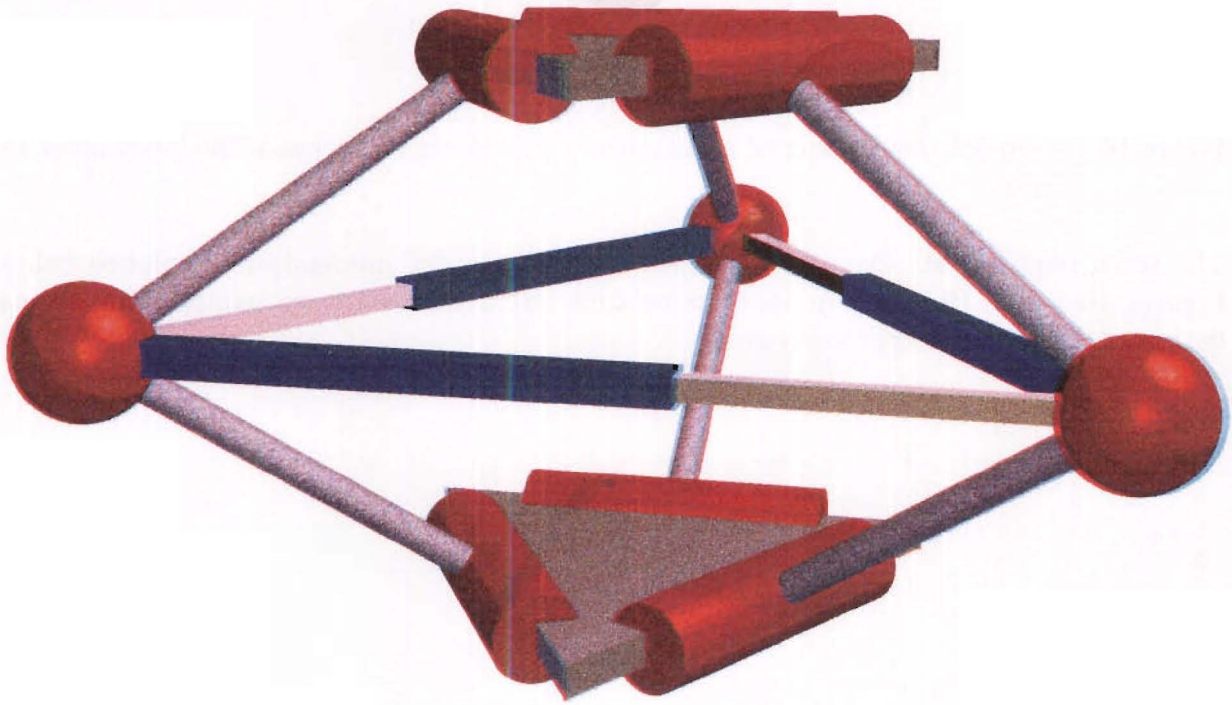


Figure 10.8 Three dof VGT (shown without counter balancing of the lower arms).

The servo requirements for a horizon sweep are plotted in figure 10.9. The very high loadings on the actuators at angles ϕ_{3a} = 60°, 180° and 300° prohibit this mechanism from being used for beam aiming applications where low elevations angles are required. These high forces occur when two of the lower/upper arms of the mechanism become close to planar with the base/platform as shown in figure 10.10. In fact if the arms were to become truly planar with

the platform, the mechanism would be singular and the actuator forces would tend to infinity. This is a singularity of the second kind (see Gosselin and Angeles [1990] for classification of singularities). In the particular case shown, the arms lie about 8° from the plane of the platform.

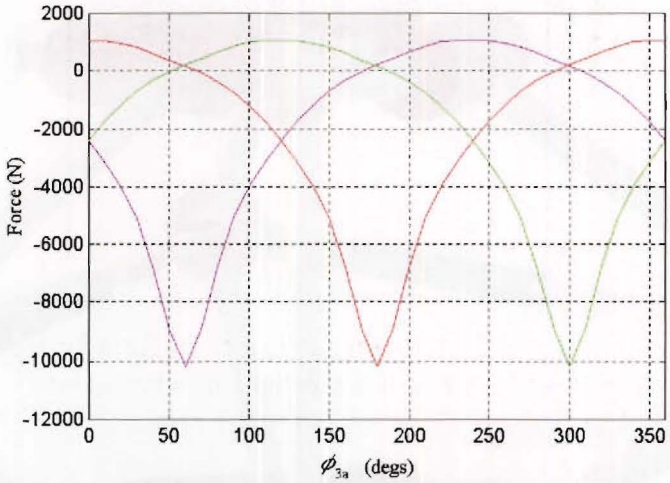


Figure 10.9 Actuator requirements for a horizon sweep of the three dof VGT with no counter balancing.

Another drawback with this mechanism is that the prismatic actuators have to undergo large changes in relative length for the entire upper hemispherical work space to realised.

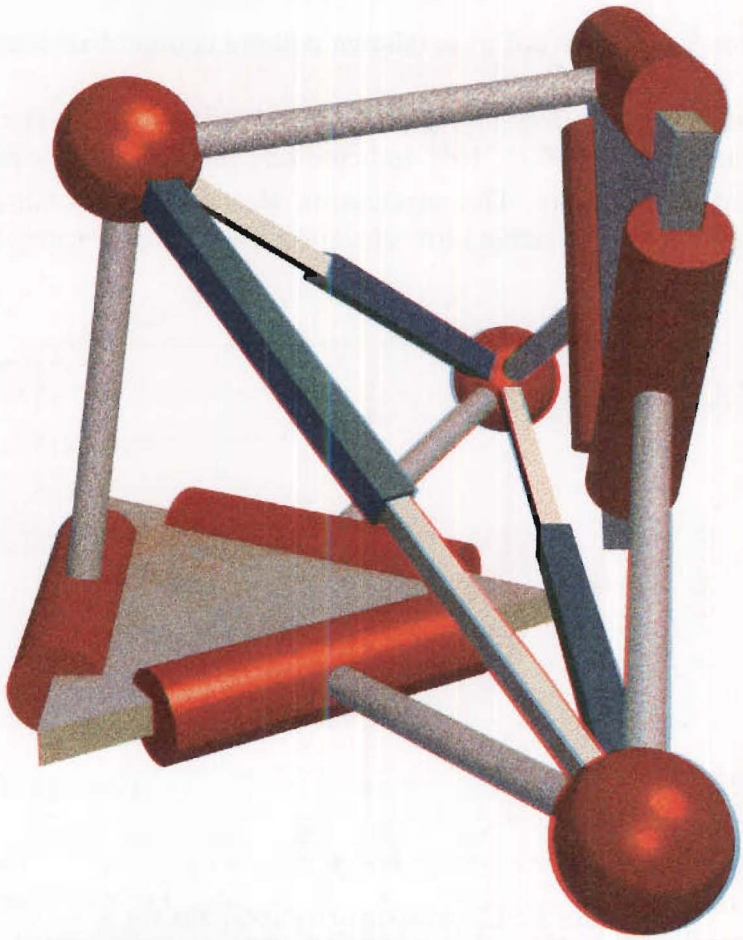


Figure 10.10 Three dof VGT in a position showing two upper arms of the mechanism close to being planar with the platform.

10.2.4 Servo requirements for the two dof truss

This two dof mechanism is similar to its three dof counter part with the exception of the removal of a prismatic actuator and the introduction of a centre strut.

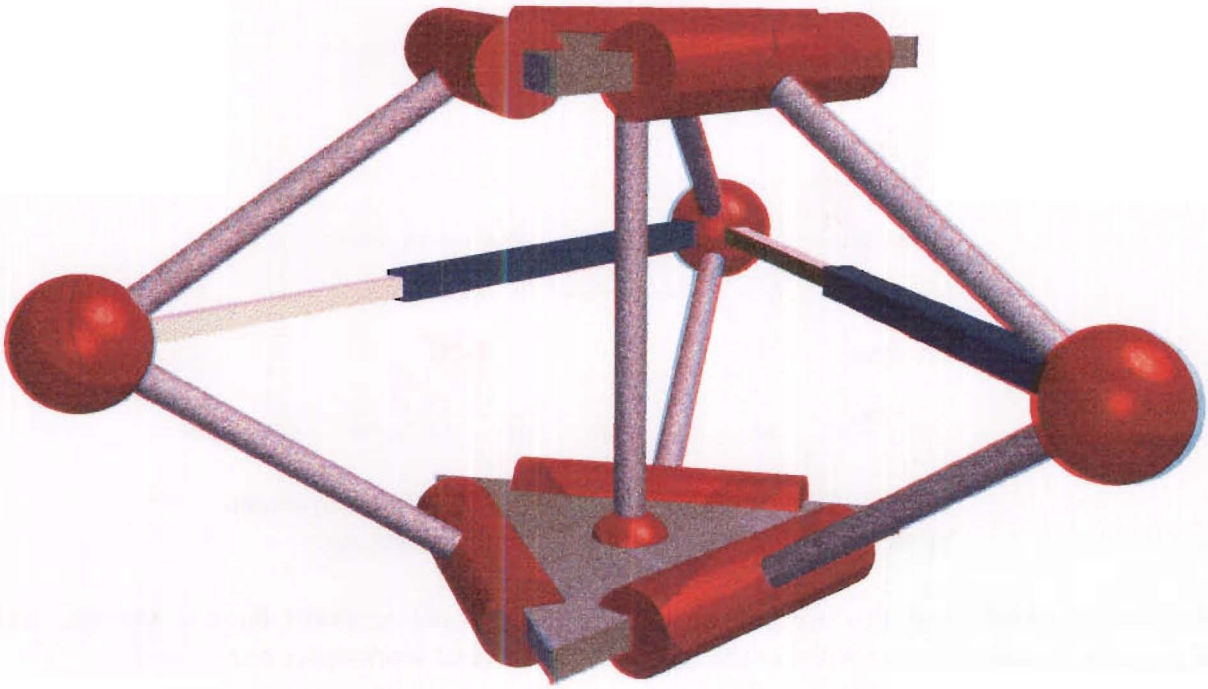


Figure 10.11 Two dof truss (shown without counter balancing).

The servo requirements for a horizon sweep are plotted in figure 10.12. The very high loadings on the actuators at angles $\phi_{3a} \approx 77°$, 163° indicate that the simulated mechanism has passed through or very near a singularity. This mechanism also still suffers similar problems to its three dof counter part when approaching low elevation angles at $\phi_{3a} = 60°$, 180° and 300°.

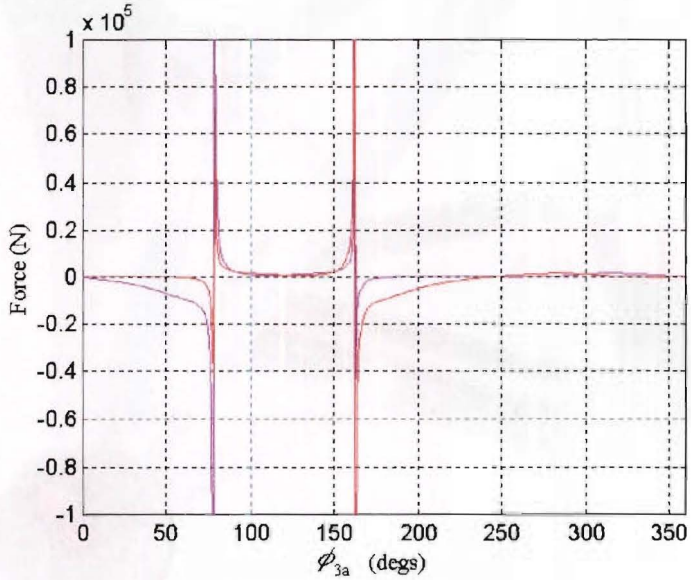


Figure 10.12 Actuator requirements for a horizon sweep of the two dof truss with no counter balancing.

A singular position is shown below in figure 10.13. When in or very close to a singularity, the platform traces out two singular point lines mirrored about the vertical plane lying at $\phi_{3a} = 120°$.

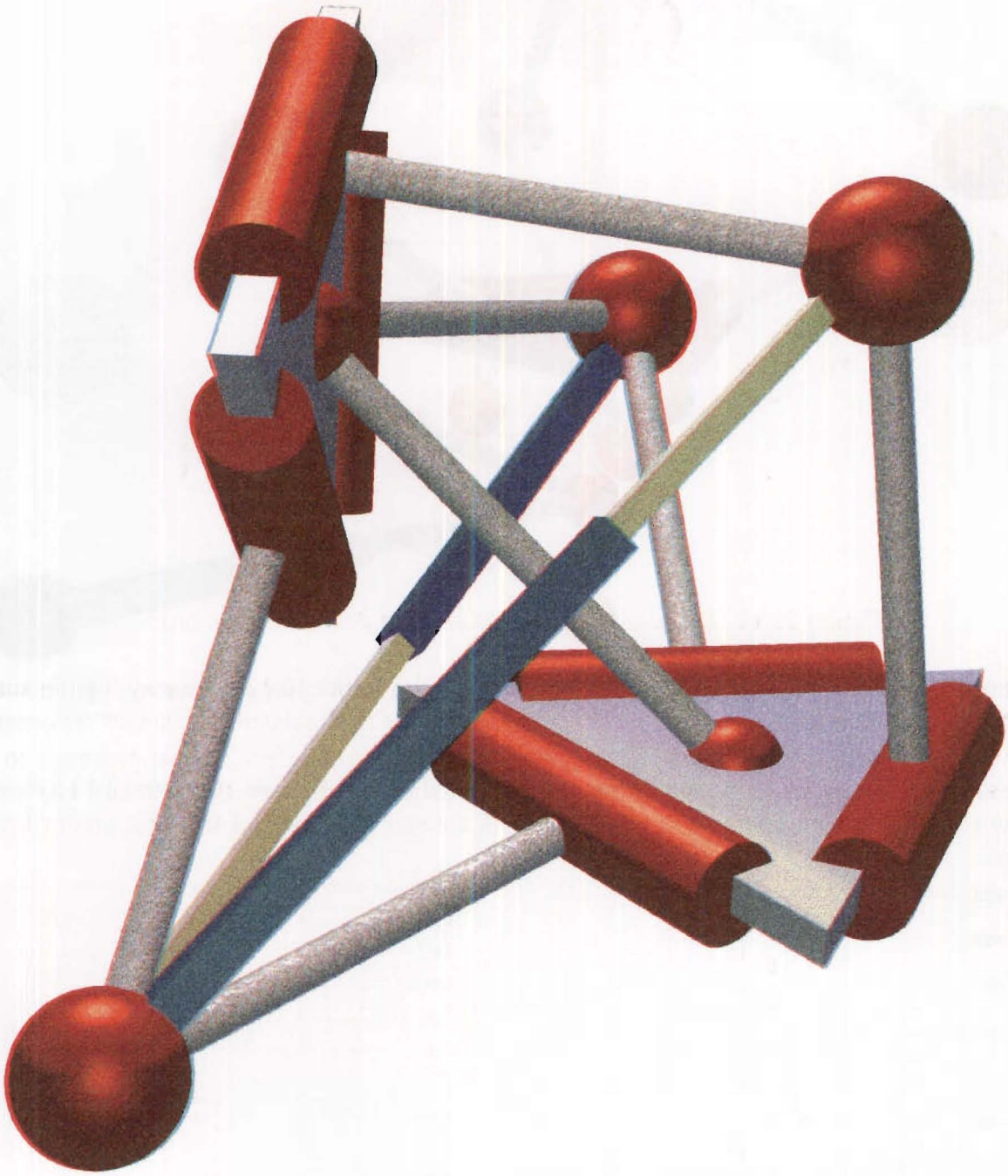


Figure 10.13 Two dof truss in a singular position.

10.2.5 Servo requirements for the two dof prismatically actuated mechanism

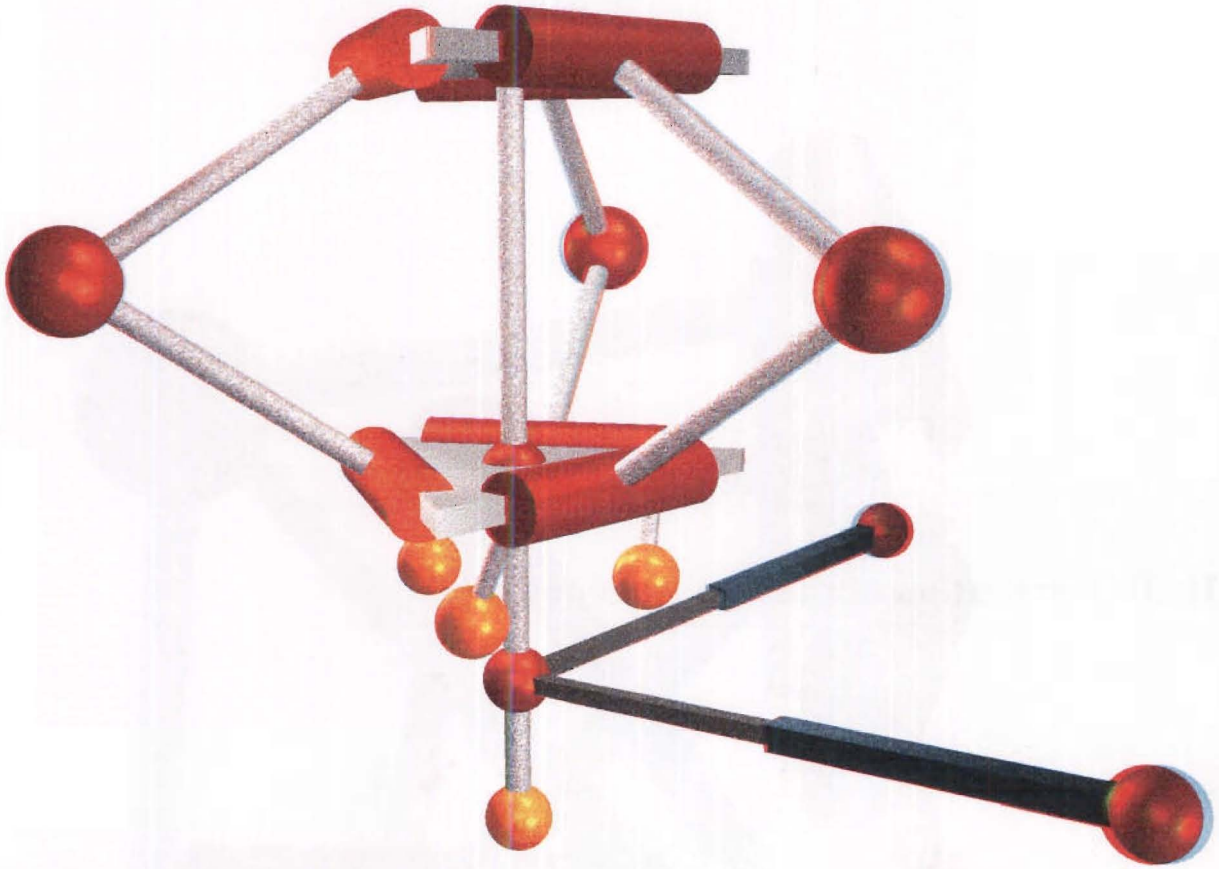


Figure 10.14 Two dof prismatically actuated mechanism with counter balancing of the lower arms and strut.

The servo requirements for a horizon sweep are plotted below. Figures 10.16 and 10.17 clearly show significant improvements to be gained with counter balancing of the strut and/or lower arms.

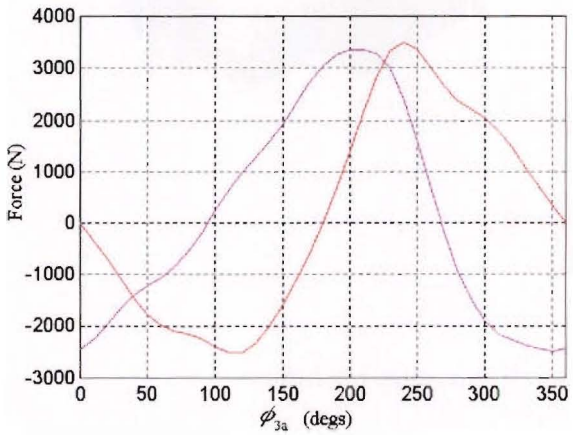


Figure 10.15 Actuator requirements for a horizon sweep of the two dof prismatically actuated mechanism with no counter balancing.

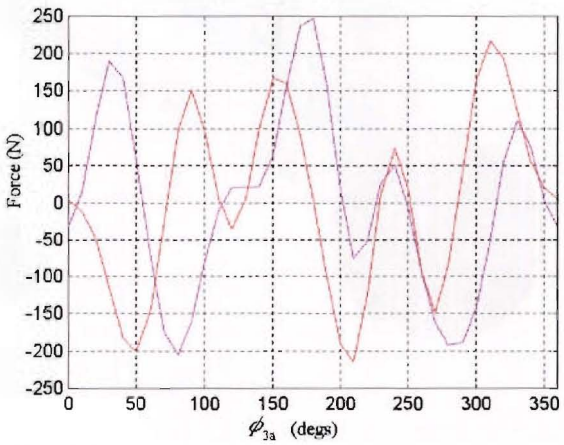


Figure 10.16. Actuator requirements for a horizon sweep of the two dof prismatically actuated mechanism with counter balancing of the strut only.

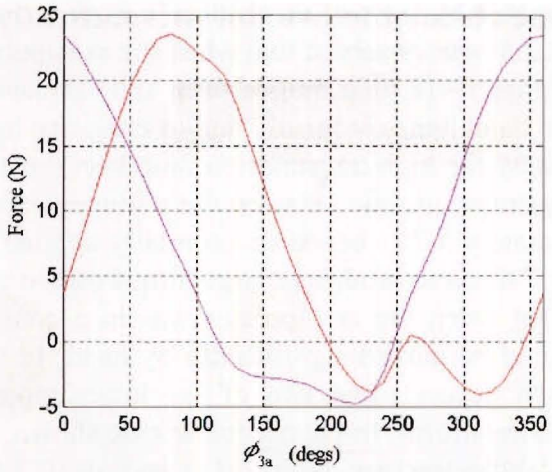


Figure 10.17 Actuator requirements for a horizon sweep of the two dof prismatically actuated mechanism with counter balancing of the strut and lower arms.

10.3 Observation of the real manipulator

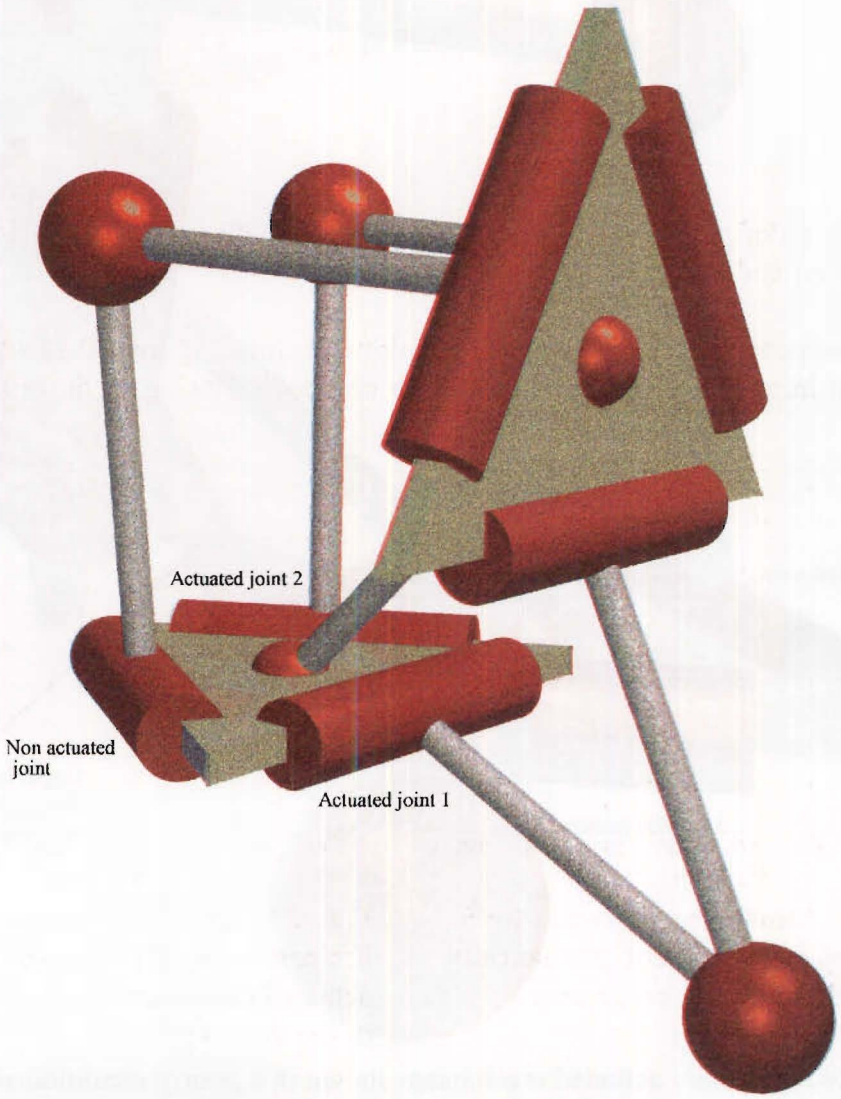


Figure 10.18 Two dof rotary actuated mechanism shown in a poorly conditioned configuration.

A problem that was highlighted by the real mechanism was the compliance of the platform at certain low elevation angles. It was observed that when the manipulator was in a configuration similar to that shown in figure 10.18, that the platform and non actuated arm could be moved significantly by hand. This compliance is mostly due to elasticity in the links and clearance in the joints. It is exacberated by the high transmission ratio between the actuated joint 1 and the non actuated joint. The transmission ratio between these joints while in this particular position was calculated as approximately 3.7:1, hence an externally applied force on the non actuated arm would be magnified 3.7 times to produce a large torque on the actuated joint 1.

It was also observed that when the manipulator was in a configuration similar to figure 10.19 that the platform could be moved significantly by hand. In this case the mechanism is nearing a singularity which occurs when two of the lower/upper arms of the mechanism become planar with the base/platform. In the particular case shown, the arms lie about 8° from the plane of the platform. Unlike the case described in section 10.2.4 where the three dof VGT is in a similar position, a large increase in actuator demand will not occur. This is explained by the fact that the transmission ratio between the two actuated joints and passive joints remains fairly low.

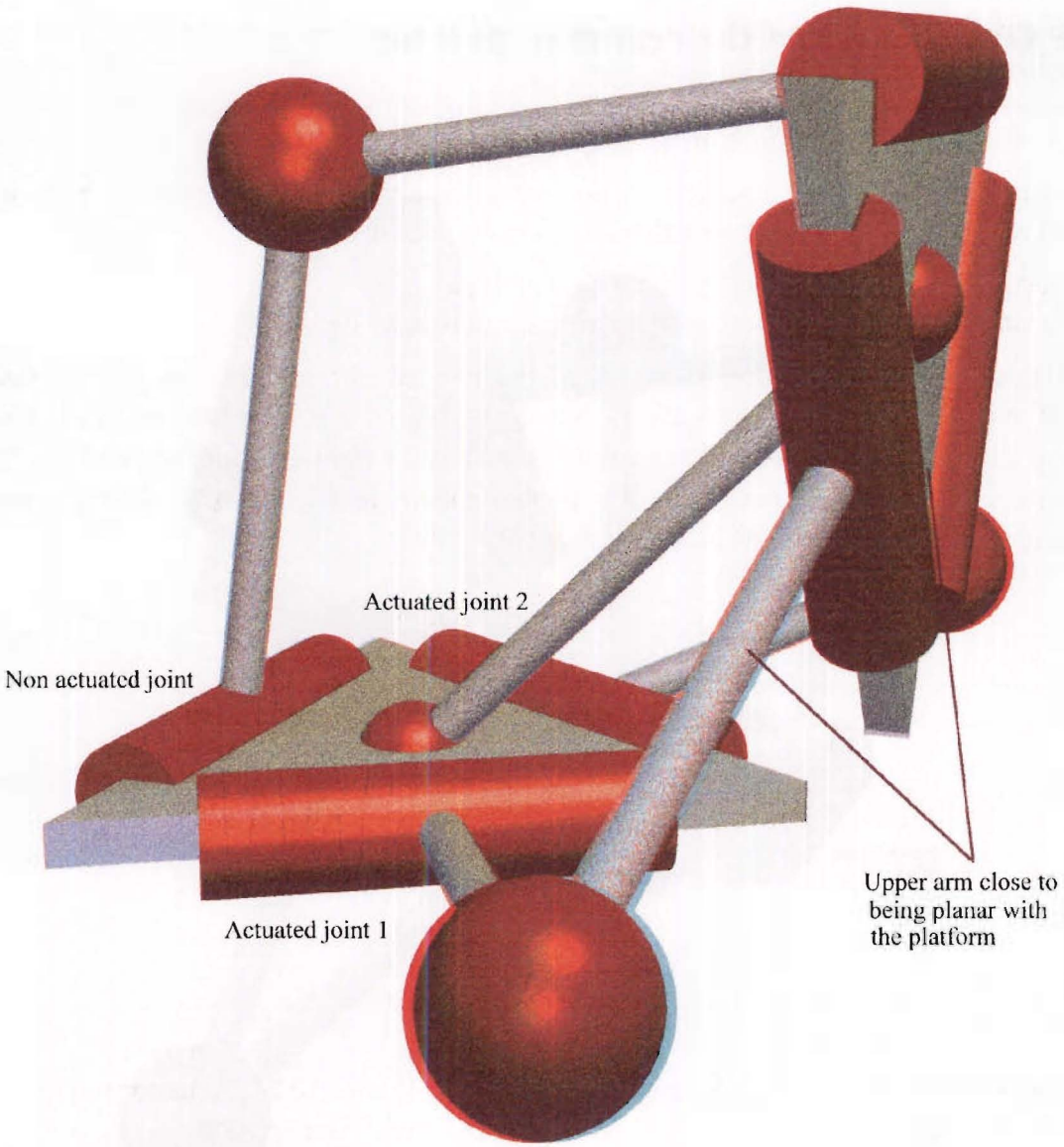


Figure 10.19 Two dof rotary actuated mechanism shown in a poorly conditioned configuration.

10.4 Maritime simulation of the two dof prismatically actuated mechanism

It is envisaged that the tracking mechanism may be used in a maritime application. If the mechanism is restricted to point above the horizon (see discussion in sec. 10.5 later), the mechanism must be mounted on a stabilised platform to ensure complete coverage of the visible hemisphere. The following numerical example is used to demonstrate the effect of the ship's motion on the antenna tracking mechanism. The rotational and translational ship motion disturbance values were taken from CCIR [1978] and Johnson [1978] respectively, and are typical of an unstabilised ship of 10,000 tons.

Disturbance input specification

Roll	±15°	10 sec
Pitch	±7.5°	7 sec
Heave	±0.25g	period not given but 10 sec used in this example

Distance between antenna and centre of rotation 76 m

The other rotary and translatory motions are not used in this example as they are typically not oscillatory.

10.4.1 Comparison of actuator requirements

The actuator requirements for the two dof prismatically actuated mechanism with a 100kg antenna shown in figure 10.14 are calculated for two scenarios below:

1) for a stationary tracked object with no ship motion.
2) for a stationary tracked object with the ship motion described above.

For this simulation the aiming angles of the tracked object w.r.t. the inertial frame are $\phi_{3a} = 0°$ and $\phi_2 = 90°$ (i.e. 0° elevation). Since a stabilised platform has been introduced, the pointing angles of the platform θ_{3a} and θ_2 are equal to the aiming angles ϕ_{3a} and ϕ_2. Thus the mechanism is in a configuration similar to that shown in figure 10.18. In the simulations below, counter balancing of the strut only is incorporated.

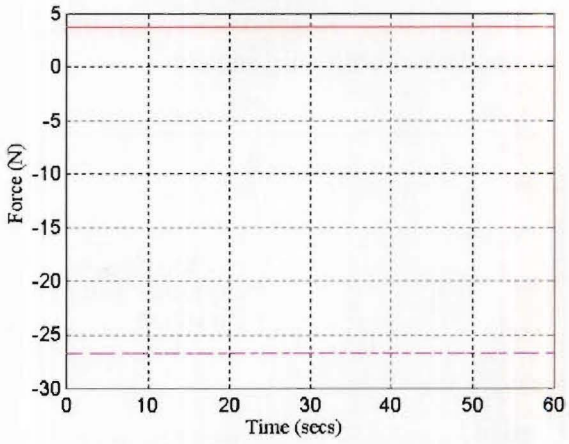


Figure 10.20 Actuator requirements without ship motion.

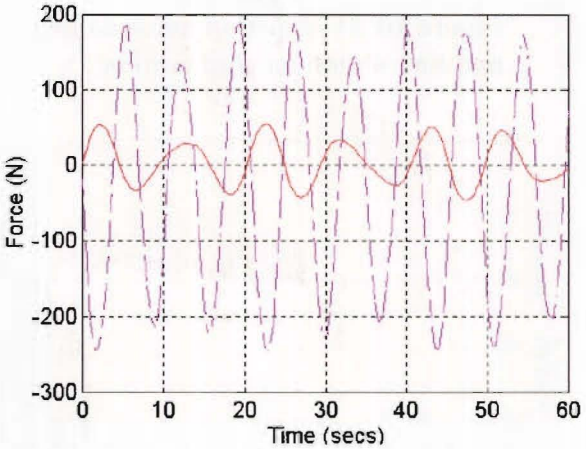


Figure 10.21 Actuator requirements with ship motion.

10.4.2 Link reactions

Of importance to the designer is the effect of the ship motion on the reaction forces experienced by the links. The plots below demonstrate the significant increase in the magnitudes of the moments and reaction forces between the platform (link 3) and neighbouring links 2,4,8 and 12, which are the upper arms and strut respectively (c.f. figure 9.2).

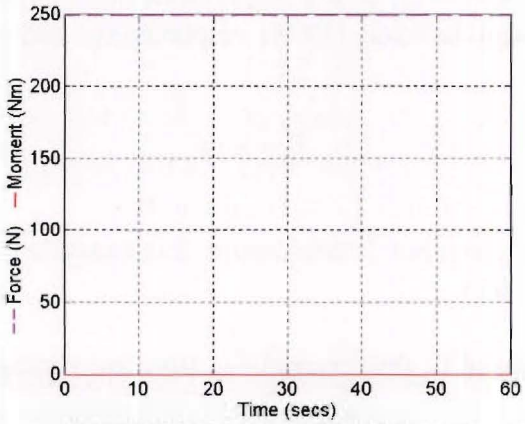


Figure 10.22 Reaction between link 2 and link 3 without ship motion.

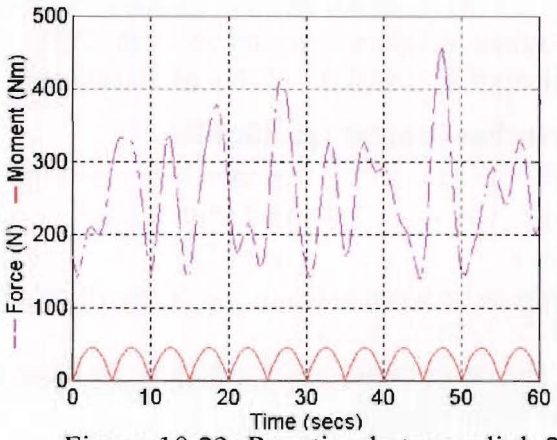


Figure 10.23 Reaction between link 2 and link 3 with ship motion.

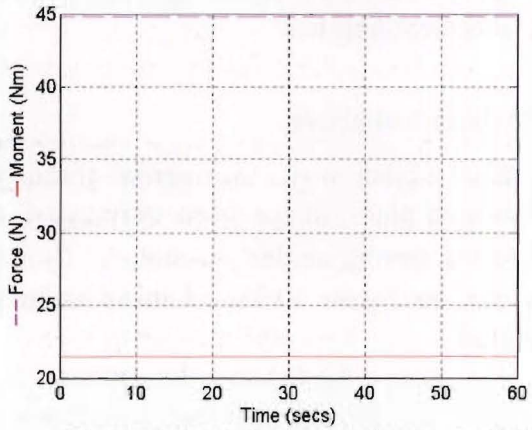


Figure 10.24 Reaction between link 3 and link 4 without ship motion.

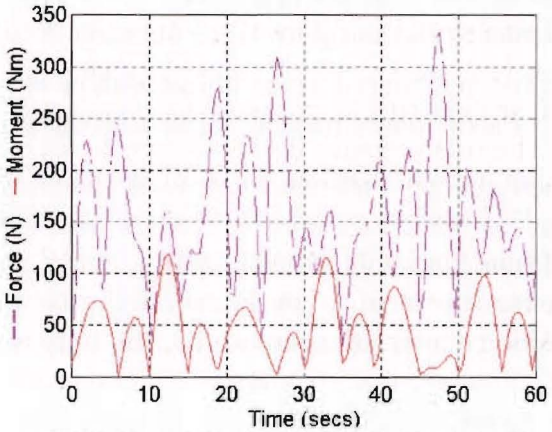


Figure 10.25 Reaction between link 3 and link 4 with ship motion.

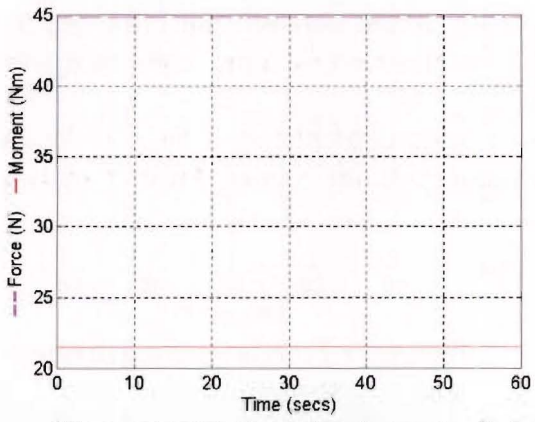


Figure 10.26 Reaction between link 3 and link 8 without ship motion.

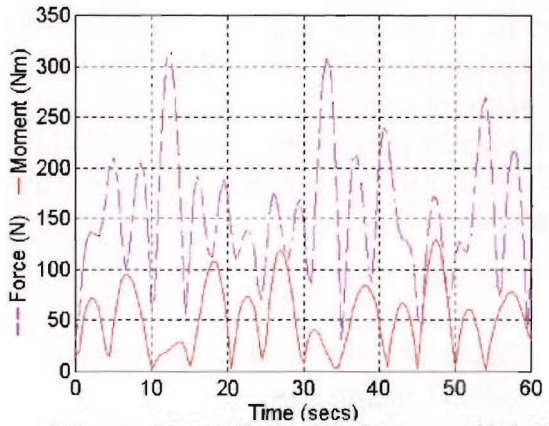


Figure 10.27 Reaction between link 3 and link 8 with ship motion.

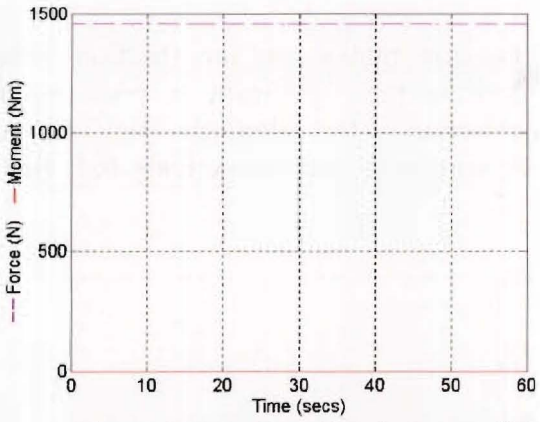


Figure 10.28 Reaction between link 3 and link 12 without ship motion.

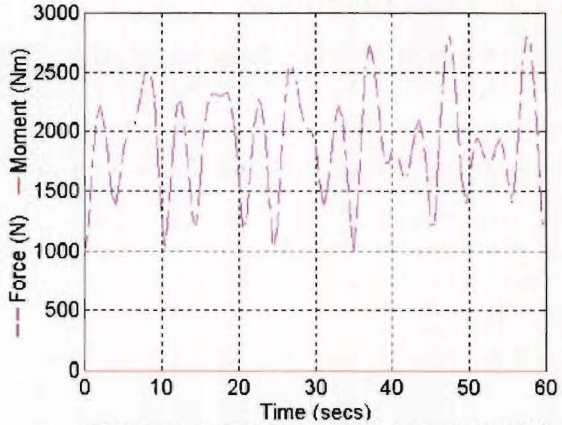


Figure 10.29 Reaction between link 3 and link 12 with ship motion.

10.5 Discussion

A number of different rotary and prismatically actuated arrangements have been investigated with and without counter balancing. The three dof mechanism was investigated for completeness but the emphasis was on the two dof mechanism employed as an antenna tracker which requires only two degrees-of-freedom. It was found that the prismatically actuated VGT system had high static loadings, especially when then mechanism approached singularities. These singular positions were found to be well inside the working hemisphere for the two dof truss (c.f. section 10.2.4) and close enough to the working hemisphere for the three dof VGT (c.f. section 10.2.3) to cause very high static loadings.

The rotary actuated two dof mechanism was shown to have very low torque requirements when counter balancing was used. Unfortunately upon observing the real robot, it became obvious that compliance in the links and clearance in the joints resulted in the stiffness of the antenna look angles being quite low in certain pointing directions (c.f. section 10.3). In a private communication, Hunt [1973,1978] pointed out that while the mechanism is in the position shown in figure 10.18, the transmission angle is quite low and may drop to around 30°. The two dof prismatically actuated mechanism (c.f. figure 10.14) may be more suitable for antenna tracking since the transmission angle between the driving links and strut is always equal to or greater than 45°.

The two dof prismatically actuated mechanism was subject to a ship mounted simulation with a stabilised platform and a set of results was shown to illustrate the effect of the ship's motion. There was a significant increase in the reaction forces between the links due to the motion of the ship which will need to be taken into account when designing a mounting system for maritime use.

It may be possible to do away with the stabilised platform completely if the compliance of the mechanism is satisfactory at low negative elevation pointing angles. Further analysis is needed in this area.

Chapter 11

Conclusions

A number of conclusions and recommendations for future research are drawn from both the mathematical modelling of the mechanisms and the real robot built at the university.

Initially a number of spatial mechanisms were briefly investigated to see if they could be suited to beam aiming applications. Most suffered from over constraint and/or potential interference problems and therefore were deemed not suitable for beam aiming applications. There may be, and almost certainly are alternative mechanisms that may be suited to beam aiming applications that are not presented in this thesis. Future work may entail a more rigorous search for other candidate mechanisms.

The forward and inverse kinematics were developed for the two and three dof mechanisms. Future work may entail developing more general models for the inverse kinematics for both mechanisms that do not rely on symmetry through the homo kinetic plane. A rigorous singular position analysis of the mechanisms has not been carried out in this thesis. Many singular positions are apparent from manipulating simple models but future work may entail a more detailed analysis of the singular positions for each mechanism. A static model was developed using traditional vector statics so that the more sophisticated models developed in Chapter 9 could be verified. The results obtained by both methods compared well.

Two large two and three degree of freedom robots were built at the university and both were used to demonstrate the feasibility of the mechanisms for beam aiming applications. The main problem is with platform compliance at certain low elevation angles as outlined in section 10.3. The mathematical modelling of the mechanisms in this thesis is unable to give a quantitative value for this compliance as only rigid body models were used. Future work should use flexible linked models using only the first mode (static deflections). Joint clearances could also be modelled. If these new variables are incorporated, a quantitative measure of the compliance can be gained.

Because of this platform compliance, two truss like mechanisms were investigated that were derivatives of the two rotary actuated mechanisms built at the university. It was hoped that the truss like structures would provide extra stiffness. Although this is generally true, both truss structures suffer from singularities in the upper hemispherical work space and so would be unlikely candidates for beam aiming applications unless a reduced coverage is required.

Finally a prismatically actuated mechanism based on the two dof rotary actuated mechanism was investigated. This mechanism would seem to be the most viable of the five mechanisms modelled for the following reasons.

a) It has the minimum number of freedoms (2) required to track an object.
b) The transmission angle is greater than in any of the other mechanisms.
c) Since it is prismatically actuated, no expensive rotary gear box is needed.
d) It is easily counterbalanced.

The position of the platform should be fairly rigidly set since it is effectively cantilevered by the central strut. A question remains over the compliance of the platform orientation, since

three passive chains are used to restrict the orientation freedoms. To ensure reliable communication the platform compliance must remain less than the beamwidth of the antenna for all look angles. The mechanism must be stiff enough so that wind loading and inertial effects from ship motion cannot affect communication. Flexible linked models using only the first mode (static deflections) should be developed so that the compliance of the platform can be calculated. Further design is needed to ensure that the prismatic actuators attached to the strut do not interfere with the antenna when pointing at low elevation angles.

No rigorous optimisation of the geometry has been carried out in this thesis. Future work should entail finding the optimum ratio between the link length, the strut length and base/platform size. This work would probably need to be done in conjunction with the stiffness/deflection modelling so that the best ratio can be found to minimise the compliance of the mechanism.

In conclusion, the antenna aiming keyhole problem has been overcome by means of a two dof parallel robotic system. A number of papers have been published on the work presented in this thesis and several more are in preparation. Future work is needed ensure that the compliance of the mechanism can be reduced to an acceptable level and to optimise the system so that it can be developed through to commercial production.

Bibliography

Afzulpurkar N. V., Ma Li, Dunlop G. R. & Johnson G. R., "Design of a parallel link robot", *Proc. NELCON'88*, Christchurch, New Zealand, pp. 52-58, (1988).

Afzulpurkar N. V., "Kinematics, design, programming and control of a robotic platform for satellite tracking and other applications", *Ph.D thesis*, University of Canterbury, New Zealand. (1990).

Alizade R., Tagiyev N. and Duffy J., "A Forward and Reverse Displacement Analysis of an In-Parallel Spherical Manipulator", *Mechanisms and Machine Theory*, Vol. 29, No. 1, pp. 125-137, (1994).

Behi F., "Kinematic Analysis for a Six-Degree-of-Freedom 3-PRPS Parallel Mechanism", *IEEE J. of Robotics and Automation*, Vol. 4, No. 5, pp. 561-565, (1988).

Burnside W. S. and Panton A. W., *Theory of equations: with an introduction to the theory of binary algebraic forms*, Vol 2, p. 74. Dublin University Press, (1889).

CCIR, *Mobile Services. Recommendations and reports of the CCIR,* XIVth plenary assembly, Kyoto, Volume VIII, Rep. 594-1, pp. 381-399, (1978).

Charentus S. and Renaud M., "Modelling and Control of a Modular, Redundant Robot Manipulator", *First International Symposium on Experimental Robotics*, Montreal, Canada, June 19 - 21, (1989).

Chen Ning-Xin and Song Shin-Min., "Direct Position Analysis of the 4-6 Stewart Platforms", *ASME Journal of Mechanical Design*, Vol. 116, pp. 61-66, (1994).

Crawford P. S. and Brush R. J. H., "Trajectory optimisation to minimise pointing error", *IEE Computing & Control Engineering Journal*, Vol 6, No. 2, pp. 61-67, (1995).

Dang N, D., Pike D., Claydon B. and Crilley P., "A stabilised Satellite Terminal for Hostile marine Environments," *Proc., 3rd European Conference on Satellite Communications*, pp. 112-116, (1993).

Dasgupta Bhaskar and Mruthyunjaya T. S., "A Canonical Formulation of the Direct Position Kinematics Problem for a General 6-6 Stewart Platform," *Mechanisms and Machine Theory*, Vol. 29, No. 6, pp. 819-827, (1994).

D'Eleuterio G. M. T., Hughes P. C. and Sincarsin G. B., "Recursive Dynamics of Elastic Chains", *Dynacon report SS-103*, Dynacon Enterprises Ltd, (1988).

Dunlop G. R., "Communications systems for remote construction sites", *Proc. First IES Information Technology Conference*, Singapore, Vol. 1, pp. 178-184, (1989).

Dunlop G. R. & Afzulpurkar N. V., "Six degree of freedom parallel link robotic mechanism: Geometrical design considerations", *Proc. IMC Conf*, Chch, NZ, pp. 291-298, (1988).

Dunlop G. R., Lintott A. B. and Jones T. P., "Linear and Spherical Robots with Three DOF", *Proc. First Australian Workshop on the Theory of Machines & Mechanisms*, Melbourne University Press, ISBN 0-7325-0594-1, pp. 175-201, (1992).

Dunlop G. R., Lintott A. B. and Jones T. P., "Three DOF parallel robots for linear and spherical movements", *ISRAM'94, Intelligent Automation and Soft Computing*, Eds. Jamshidi M, and Nguyen CC, TSI Press, ISBN 0-962-7451-4-6, Vol 1, pp. 655-660, (1994).

Ellis P. J., Tracking Antenna Mount. *European Patent Application* No. 87306732.6, (1987).

Fichter E. F., "A Stewart Platform Based Manipulator: General Theory and Practical Construction", *The International Journal of Robotics Research*, Vol. 5,No. 2, pp. 157-182, (1986).

Fichter E. F. & McDowel J., "A novel design for a robot arm", *Proc. Int. Computer Tech. Conf.*, New York, ASME, pp. 250-256, (1980).

Fuan Wen and Chonggao Liang, "Displacement Analysis of the 6-6 Stewart Platform Mechanisms", *Mechanisms and Machine Theory*, Vol. 29, No. 4, pp. 547-557, (1994).

García de Jalón J. and Bayo E., *Kinematic and dynamic simulation of multibody systems: the real time challenge*, Springer-Verlag, New York, Inc, (1994).

Gosselin C. and Angeles J., "The Optimum Kinematic Design of a Spherical Three-Degree-of-Freedom Parallel Manipulator," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 111, pp. 202-207, (1989).

Gosselin C. and Lavoie E., "On the Kinematic Design of Spherical Three-Degree-of-Freedom Parallel Manipulators", *The International Journal of Robotics Research*, Vol. 12, No. 4, pp. 394-402, (1993).

Gosselin C. and Sefrioui J., "Solution Polynomiale au Probleme Geometrique Directe d'un Manipulateur Parallele Spherique", *Thirteenth Canadian Congress of Applied Mechanics*, June 2-6, pp. 670-671, (1991).

Hertz R. B. and Hughes P. C., *Computational Kinematics*, p. 241-250 J. Angles et al. (eds), (1993).

Huang S., Natori M. C. and Miura K., "Motion Control of Free-Floating Variable Geometry Truss Part1: Kinematics", *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, (1996).

Hughes P. C., *Spacecraft Attitude Dynamics*, John Wiley & Sons, Inc, (1986).

Hughes P. C., "Multibody dynamics for space station manipulators: Dynamics of a chain of elastic bodies", *Dynacon report SS-2*, Dynacon Enterprises Ltd, (1985).

Hughes P. C. and Sincarsin G. B., "Dynamics of an elastic multibody chain: Part B-Global dynamics", *Dynamics and Stability of Systems*, Vol 4, Nos. 3&4, (1989).

Hunt K. H., *Kinematic Geometry of Mechanisms*, Oxford University Press, London, (1978).

Hunt K. H., "Constant-Velocity Shaft Couplings: A General Theory", *Journal of Engineering for Industry*, pp. 455-464, (1973).

Hunt K. H. and Pimrose E. J. F., "Assembly Configurations of Some In-Parallel-Actuated Manipulators", *Mechanisms and Machine Theory*, Vol. 28, No. 1, pp. 31-42, (1993).

Husain M. and Waldron K. J., "Position Kinematics of a Three-Limbed Mixed Mechanism", *ASME Journal of Mechanical Design*, Vol. 116, pp. 924-929, (1994).

Johnson M. B., "Antenna control system for a ship terminal for marisat", *Proc. Maritime and Aeronautical Satellite Communication and Navigation*, IEE Conference, (1978).

Kirby R. J., "A simple stabilised antenna platform for maritime satellite communications", *IEE Conference Pub.* No. 95, pp. 135-139, (1973).

Kleinfinger J. F. and Khalil W., "Dynamic modelling of closed-loop robots", *Proc. 16th International Symposium on Industrial Robots*, pp. 401-412, (1986).

Kosuge K., Takeo K., Fukuda T., Kai K., Mizuno T. and Tomimatsu H., "Computation of Parallel Link Manipulator Dynamics", *Proc. of the International Conference on Industrial Electronics, Control, and Instrumentation*, Vol. 3, pp. 1672-1677, (1993).

Kurtz R. and Haywood V., "Multiple-Goal Kinematic Optimisation of a Parallel Spherical Mechanism with Actuator Redundancy", *IEEE Journal of Robotics and Automation*, Vol. 8, No. 5, pp. 644-651, (1992).

Lambert M., *United States Patent*, No. 4,651,589, (1987).

Larochelle P. M., "Design of 3 dof Spherical Robotic Mechanisms", *IFToMM, Proc. Ninth World Congress on The Theory of Machines and Mechanisms*, Politecnico di Milano, Italy. pp. 1826-1830, (1995).

Lee K. M. and Shah D. K., "Kinematic Analysis of a three degree of freedom in-parallel actuated manipulator," *IEEE J. of Robotics and Automation*, Vol. 4, No. 3, pp. 354-360, (1988).

Lee K. M. and Shah D. K., "Dynamic Analysis of a Three-Degrees-of-Freedom In-Parallel Actuated Manipulator," *IEEE J. of Robotics and Automation*, Vol. 4, No. 3, pp. 361-367, (1988).

Li Chang-Jin and Sankar T. S., "Fast Inverse Dynamics Computation in Real Time Robot Control", *Mechanisms and Machine Theory*, Vol. 27, No. 6, pp. 741 - 750, (1992).

Li Chang-Jin and Sankar T. S., "Development of Efficient Closed-Form Dynamic Equations for Robot Manipulators using Parallel and Perpindicular Concepts", *Mechanisms and Machine Theory*, Vol. 28, No. 2, pp. 233 - 198, (1993).

Lin Shir-Kuan., "Dynamics of the Manipulator with Closed Chains", *IEEE J. of Robotics and Automation*, Vol. 6, No. 4, pp. 496-501, (1986).

Lin Wei, Crane C. D. 111 and Duffy J., "Closed-Form Forward Displacement Analysis of the 4-5 In-Parallel Platforms", *ASME Journal of Mechanical Design*, Vol. 116, pp. 47-53, (1994).

Luh, J. Y. S. and Zheng, Y. F., "Computation of input generalised forces for robots with closed-kinematic chain mechansims", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 2, pp. 95-103, (1985).

McInnis Bayliss C. and Liu Chen-Kang Frank, "Kinematic and Dynamics in Robotics: A Tutorial Based Upon Classical Concepts of Vectorial Mechanics", *IEEE J. of Robotics and Automation*, Vol. RA-2, No. 4, pp. 181-187, (1986).

Merlet J. P., "Direct Kinematics of Parallel Manipulators", *IEEE J. of Robotics and Automation*, Vol. 9, No. 6, pp. 842-846, (1993).

Miya K., *Satellite communications technology.* KDD Engineering and Consulting Ltd., English Ed, (1981).

Mohamed M. G. & Duffy J., "A Direct Determination of the Instantaneous Kinematics of Fully Parallel Robot Manipulators", *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, pp. 226-229, (1985).

Murray J. and Lovell G., "Dynamic Modelling of Closed-Chain Robotic Manipulators and Implications for Trajectory Control", *IEEE Journal of Robotics and Automation*, Vol. 5, No. 4, pp. 522-528, (1989).

Murthy V. and Waldron K. J., "Position Kinematics of the General Lobster Arm and Its Series-Parallel Dual", *ASME Journal of Mechanical Design*, Vol. 114, pp. 406-413, (1992).

Nair R. and Maddocks J. H., "On the Forward Kinematics of Parallel Manipuators", *The International Journal of Robotics Reasearch*, Vol. 13,No. 2, pp. 171-188, (1994).

Nakamura Y. and Ghodoussi M., "A computational scheme of closed link robot dynamics derived by D'Alembert principal", *Proc. 1988 IEEE Int. Conf. on Robotics and Automation* (Philadelphia, PA, Apr. 24-29), pp. 1354-1360, (1988).

Nanua P., Waldron K. J. and Murthy V., "Direct Kinematic Solution of a Stewart Platform," *IEEE J. of Robotics and Automation*, Vol. 6, No. 4, pp. 438-443, (1990).

Nathamson F. E., *Radar Design Principles, New York*, pub. McGraw-Hill, Ch.1, (1969).

Notash L. and Podhorodeski R. P., "On the Forward Displacement Problem of Three-Branch Parallel Manipulators", *Mechanisms and Machine Theory*, Vol. 30, No. 3, pp. 391-404, (1995).

Parenti-Castelli V. and Innocenti C., "Direct Position Analysis of the Stewart Platform Mechanism," *Mechanisms and Machine Theory*, Vol. 25, No. 6, pp. 611-621, (1990).

Parenti-Castelli V. and Innocenti C., "Forward Displacement Analysis of Parallel mechanisms: Closed Form Solution of PRR-3S and PPR-3S Structures", *ASME Journal of Mechanical Design*, Vol. 114, pp. 68 - 73, (1992).

Peruzzini W. Ouellet A. and Hui R., "Design and Analysis of a Three-DoF Parallel Mechanism", *IFToMM, Proc. Ninth World Congress on The Theory of Machines and Mechanisms*, Politecnico di Milano, Italy. pp. 2141-2145, (Aug 1995).

Peters R. M., "Synthesis of a Mechanism for the Stabilisation of a Radar Antenna", *Mechanisms and Machine Theory*, Vol. 13, pp. 369-389, (1978).

Phillips J., *Freedom in machinery, an introduction to screw theory*, Cambridge University Press, ISBN 0521-23696-7, Vol.1, (1984).

Pierrot F., Fournier A. and Dauchez P., "Towards a Fully-Parallel 6 Dof Robot for High-Speed Applications", *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, pp. 1288-1293, (1991).

Shabana A. S., *Computational dynamics*, John Wiley & Sons, Inc, (1994).

Sincarsin G. B. and Hughes P. C., "Dynamics of an elastic multibody chain: Part A-Body motion equations", *Dynamics and Stability of Systems*, Vol 4, Nos. 3&4, (1989).

Sreenivasan S. V., Waldron K. J. and Nanua P., "Closed Form Direct Displacement Analysis of the 6-6 Stewart Platform", *Mechanisms and Machine Theory*, Vol. 29, No. 6, pp. 855-864, (1994).

Stewart D., "A platform with six degrees of freedom", *Proc. IMechE*, Vol. 180, No. 15, pp. 371-386, (1965).

Strang G., *Linear Algebra and its Applications*, Academic Press, New York, (1976).

Tahmasebi F. and Tsai L.-W., "Closed-Form Direct Kinematics Solution of a New Parallel Minimanipulator", *ASME Journal of Mechanical Design*, Vol. 116, pp. 1141-1147, (1994).

Wampler C., "Forward Displacement Analysis of General Six-In-Parallel SPS (Stewart) Platform Manipulators Using Soma Coordinates", North American Operations Research and Development Centre, General Motors Corporation, Publication R&D-8179, (1994).

Wang L. and Chen C., "On the Numerical Analysis of General Parallel Robotic Manipulators", *IEEE J. of Robotics and Automation*, Vol. 9, No. 3, pp. 272-285, (1993).

Wen F. and Liang C., "Displacement Analysis of the 6-6 Stewart Platform Mechanisms", *Mechanisms and Machine Theory*, Vol. 29, No. 4, pp. 547-557, (1994).

Wittenburg J., *Dynamics of Systems of Rigid Bodies*, B. G. Teubner Stuttgart, (1994).

Yang D. C. H. & Lee T. W., "Feasibility Study of a Platform Type of Robotic Manipulators from a Kinematic Viewpoint", *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 106, pp. 191-198, (1984).

Zhang Chang-de and Song Shin-Min., "Forward Kinematics of a Class of Parallel (Stewart) Platforms with Closed - Form Solutions," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, pp. 2676-2681, (1991).

Zhang Chang-de and Song Shin-Min., "Forward Position Analysis of Nearly General Stewart Platforms," *ASME Journal of Mechanical Design*, Vol. 116, pp. 54-60, (1994).

Appendix A

Engineering drawings of the robot components

The design of the components has already been discussed in Chapter 3. In this appendix the detailed engineering drawings of the components are presented.

Drawing No. A.1: Harmonic drive gearbox assembly
Drawing No. A.2: Harmonic drive gearbox components
Drawing No. A.3: Harmonic drive gearbox components
Drawing No. A.4: Harmonic drive gearbox components
Drawing No. A.5: Harmonic drive gearbox components
Drawing No. A.6: Harmonic drive gearbox components
Drawing No. A.7: Harmonic drive gearbox mounting bracket
Drawing No. A.8: Base
Drawing No. A.9: Support leg
Drawing No. A.10: Extension leg
Drawing No. A.11: Floor mount for support leg
Drawing No. A.12: Lower arm
Drawing No. A.13: Lower arm
Drawing No. A.14: Upper arm
Drawing No. A.15: Upper arm
Drawing No. A.16: Revolute joints
Drawing No. A.17: Platform
Drawing No. A.18: Platform
Drawing No. A.19: Ball joint
Drawing No. A.20: Home and limit switch mounting bracket
Drawing No. A.21: Buffer mounting brackets
Drawing No. A.22: Buffer mould

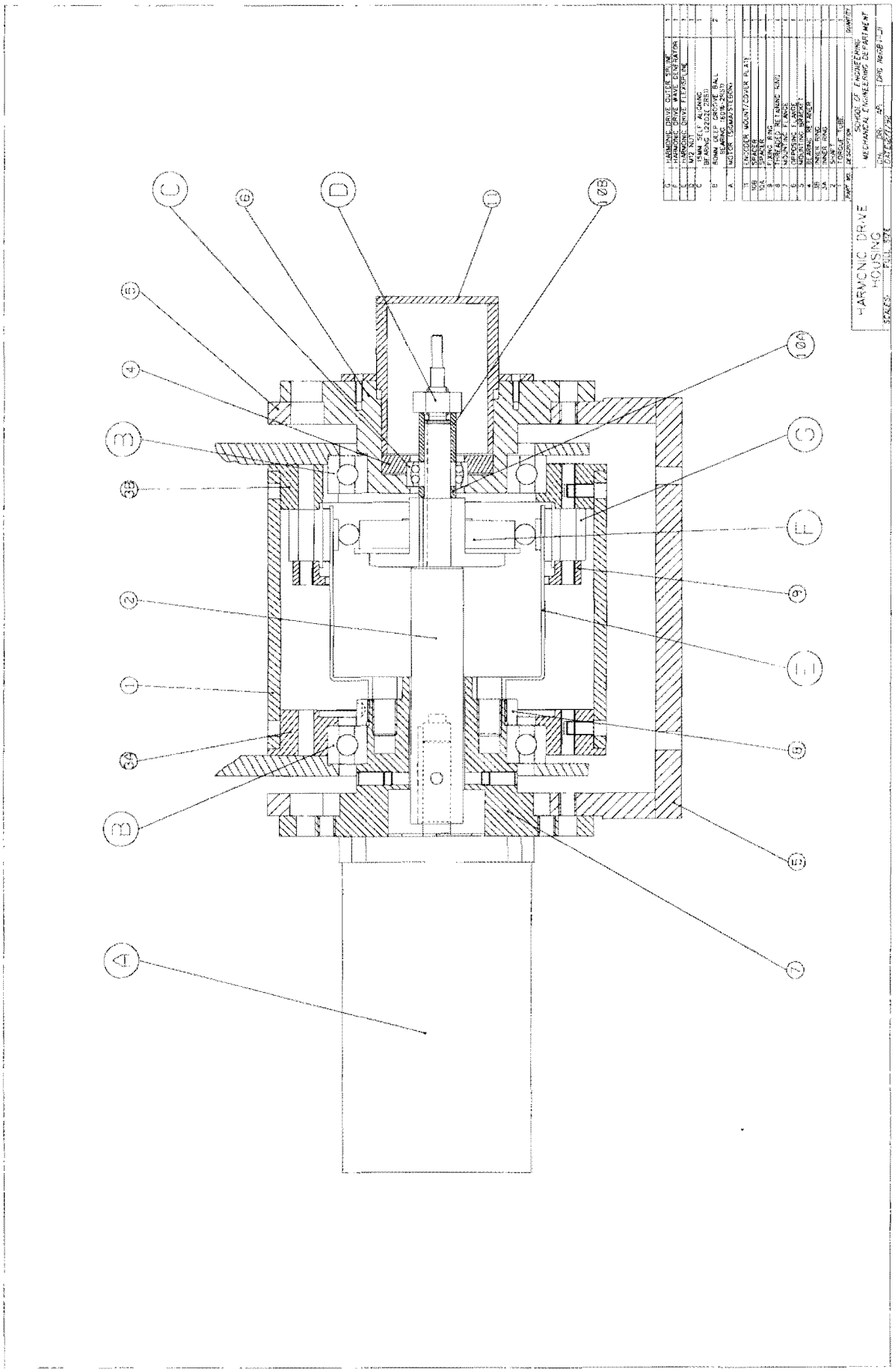


Figure A.1 Harmonic drive assembly.

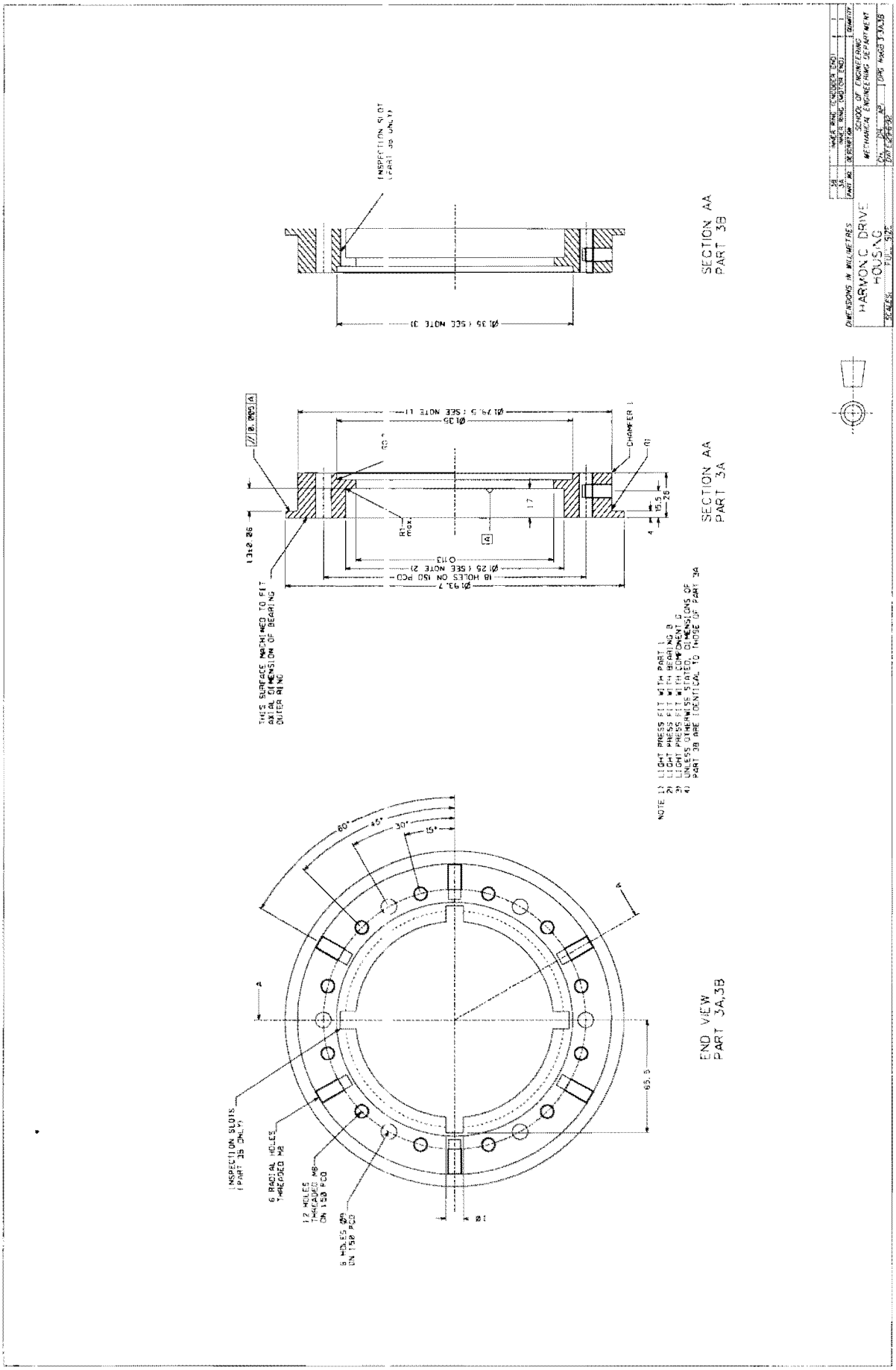


Figure A.2 Gearbox components.

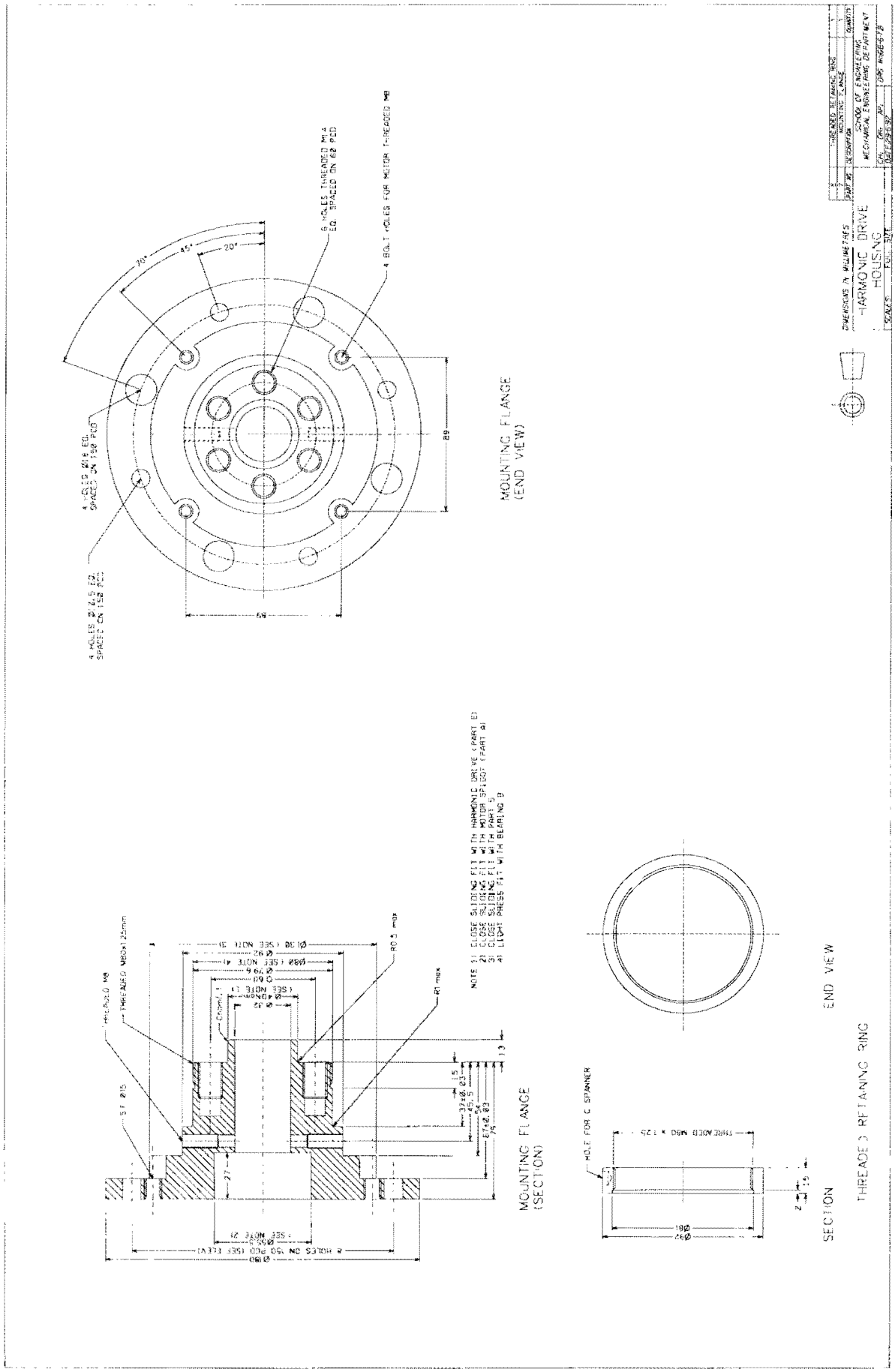


Figure A.3 Gearbox components.

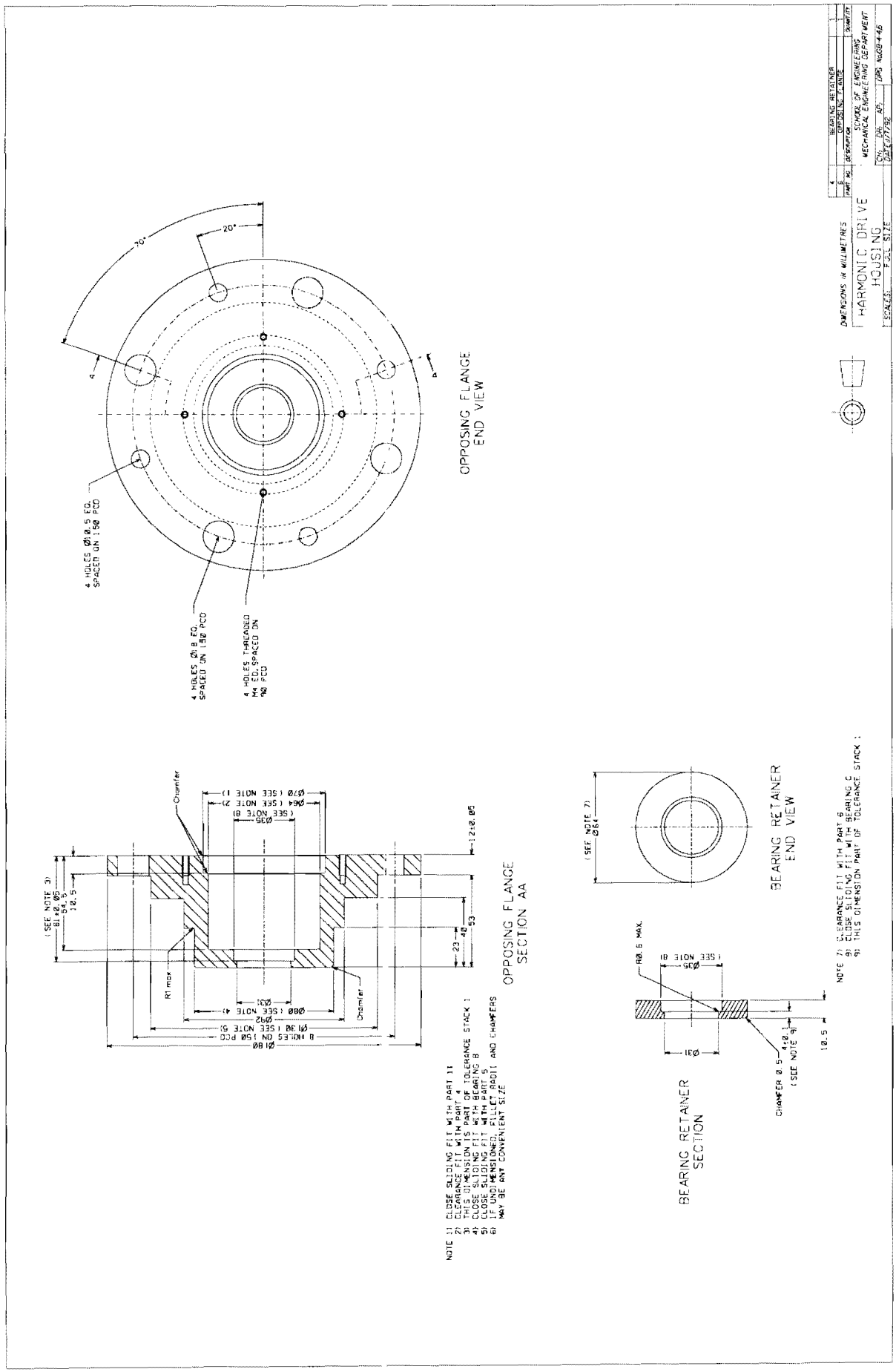


Figure A.4 Gearbox components.

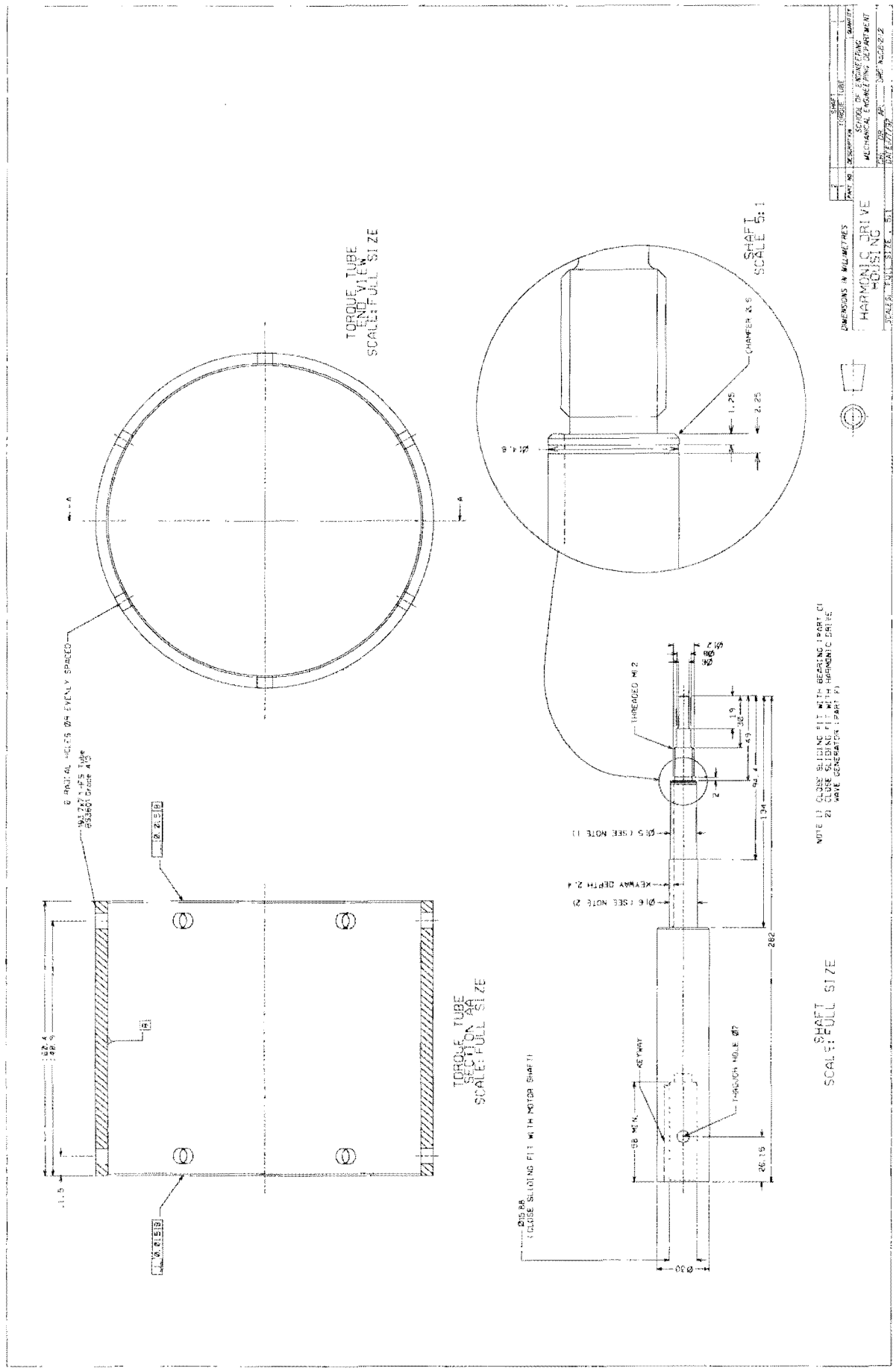


Figure A.5 Gearbox components.

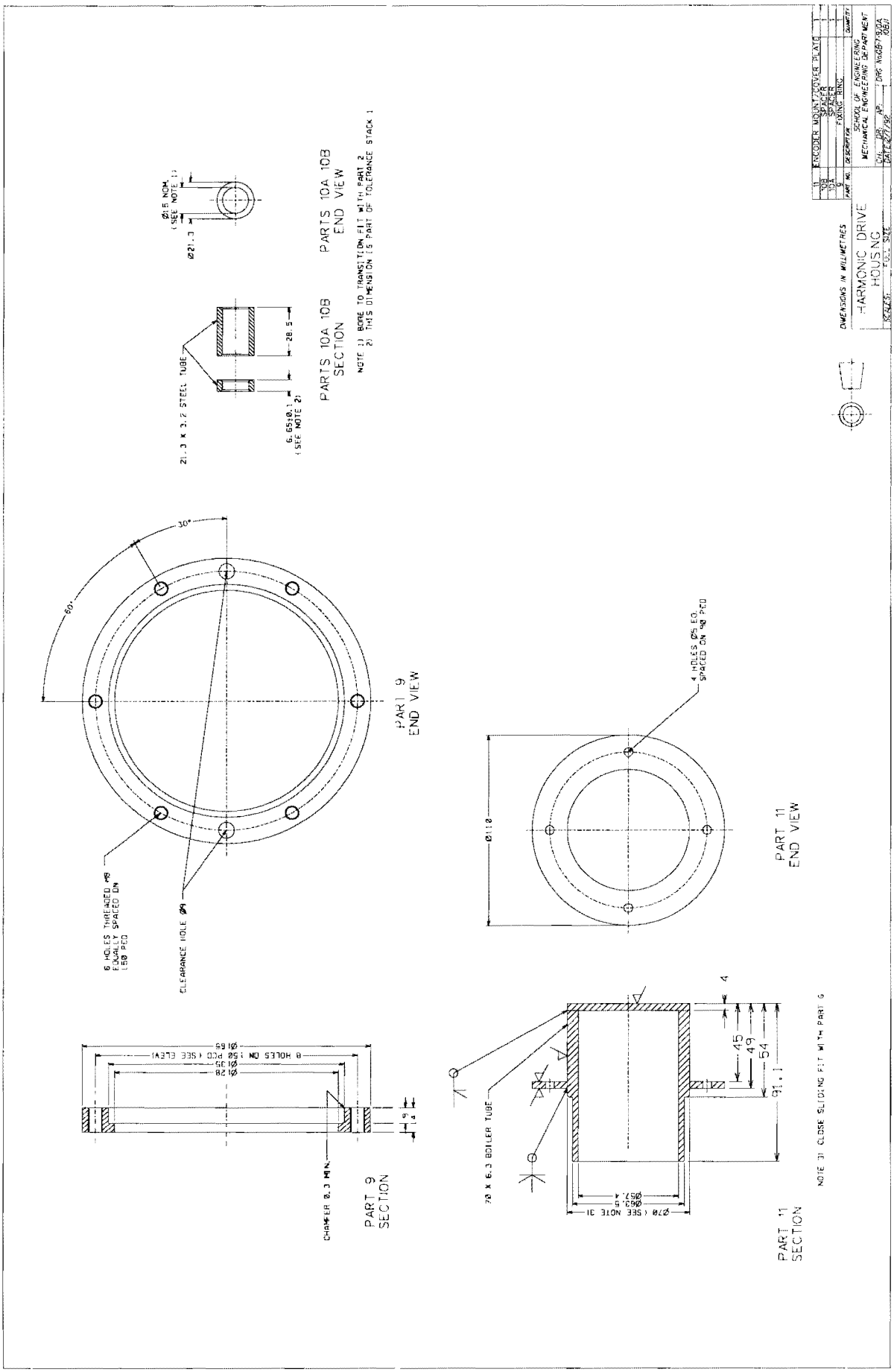


Figure A.6 Gearbox components.

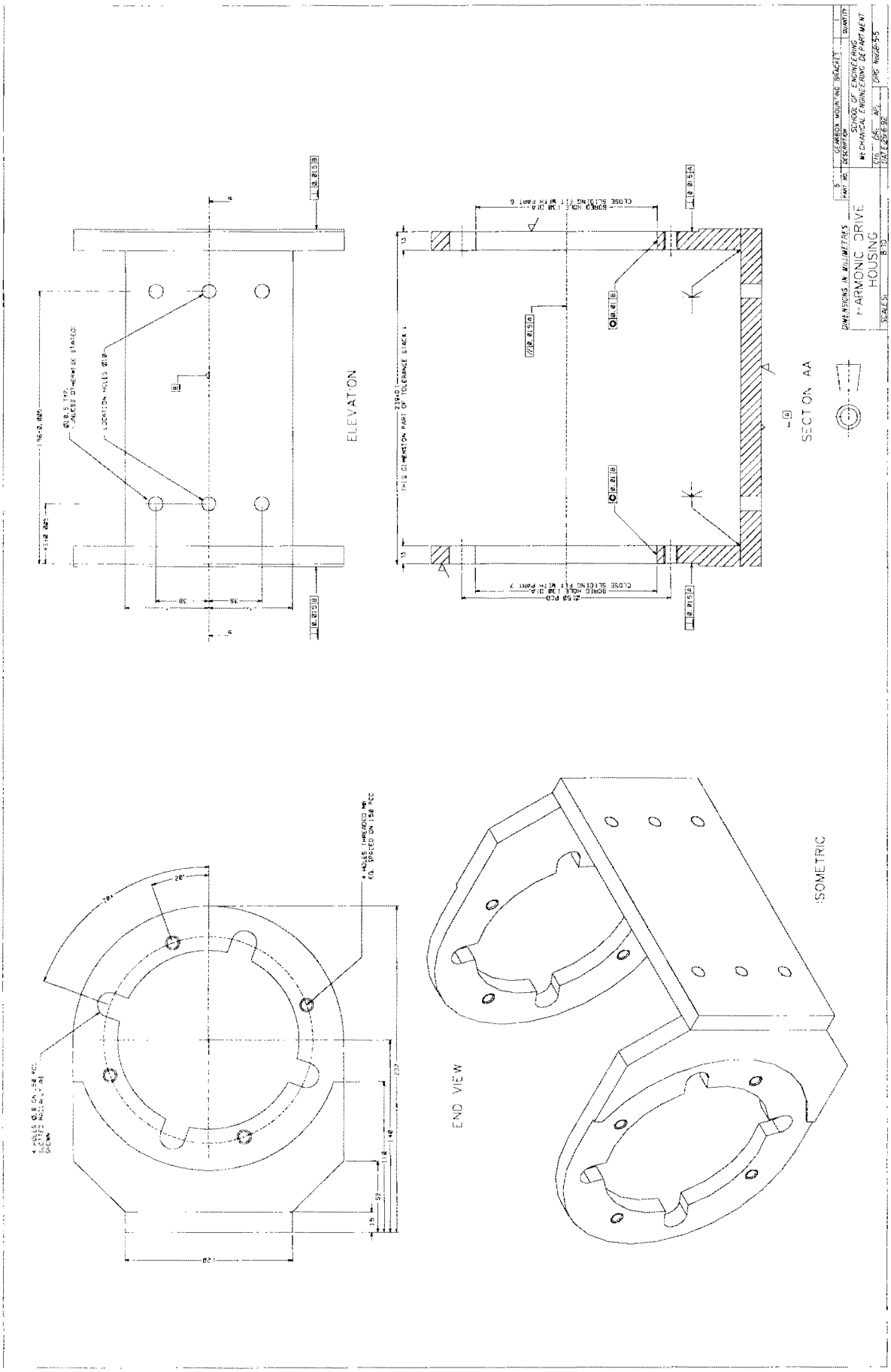


Figure A.7 Gearbox mounting bracket.

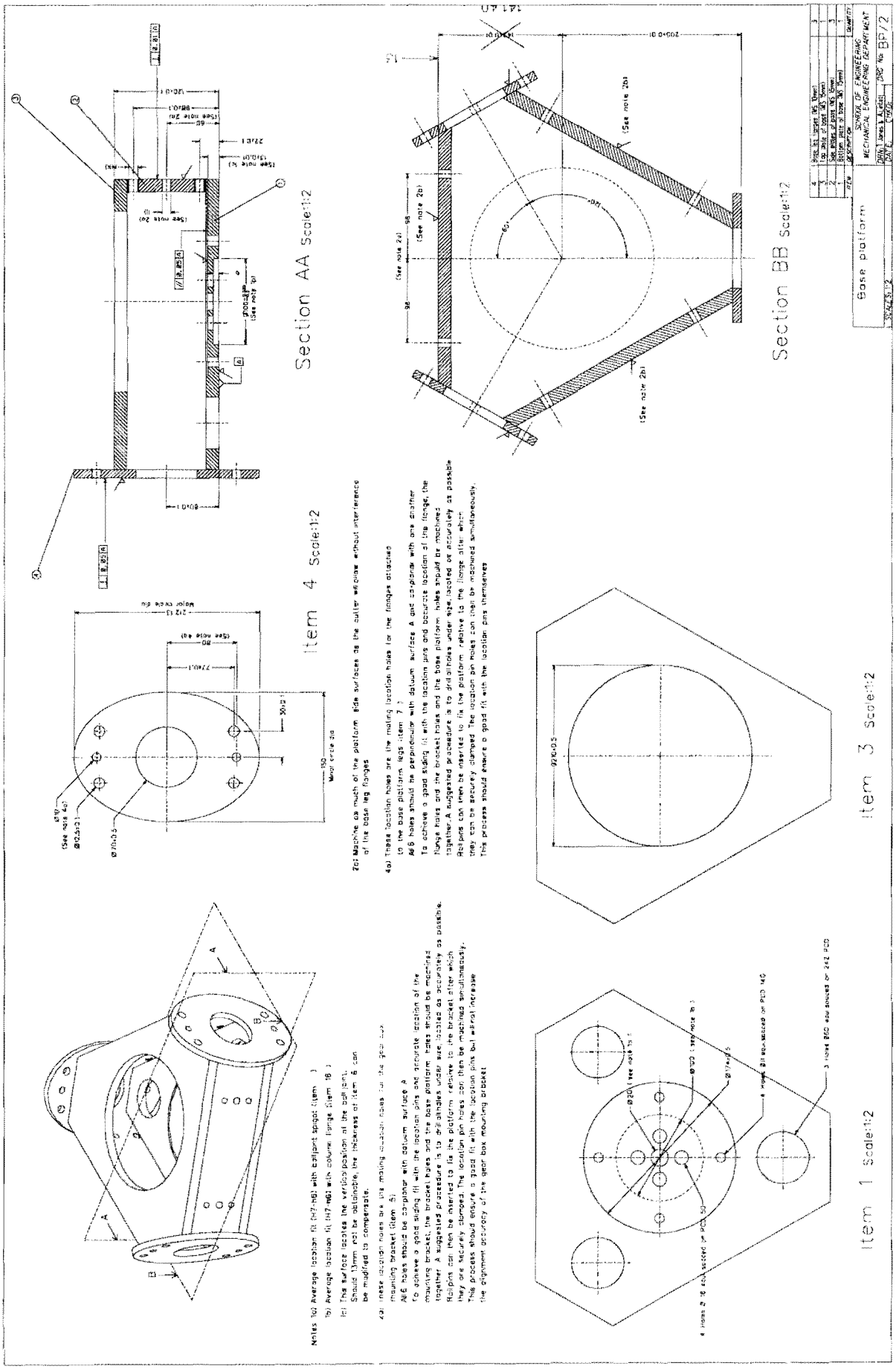


Figure A.8 Base.

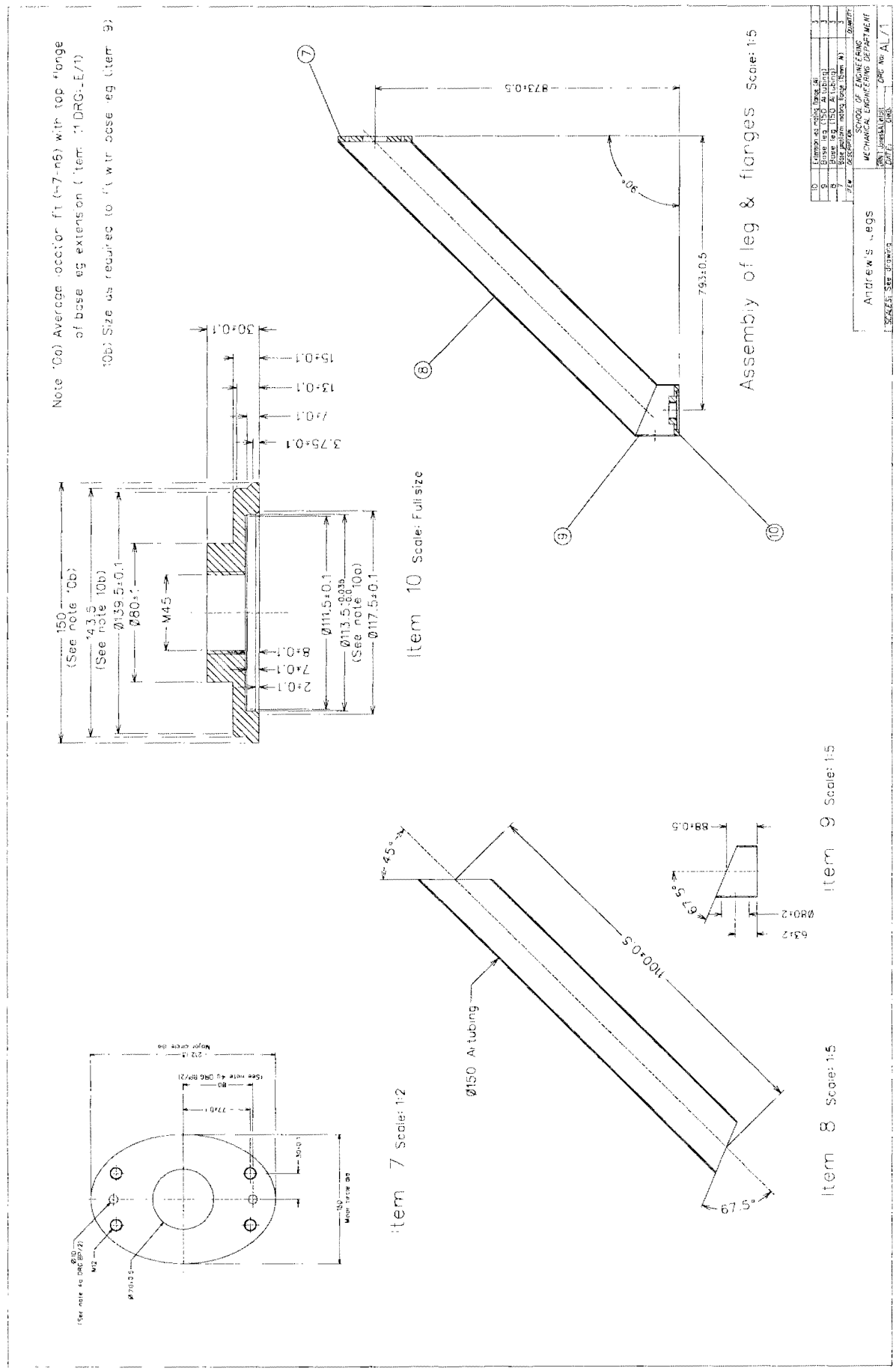


Figure A.9 Support leg.

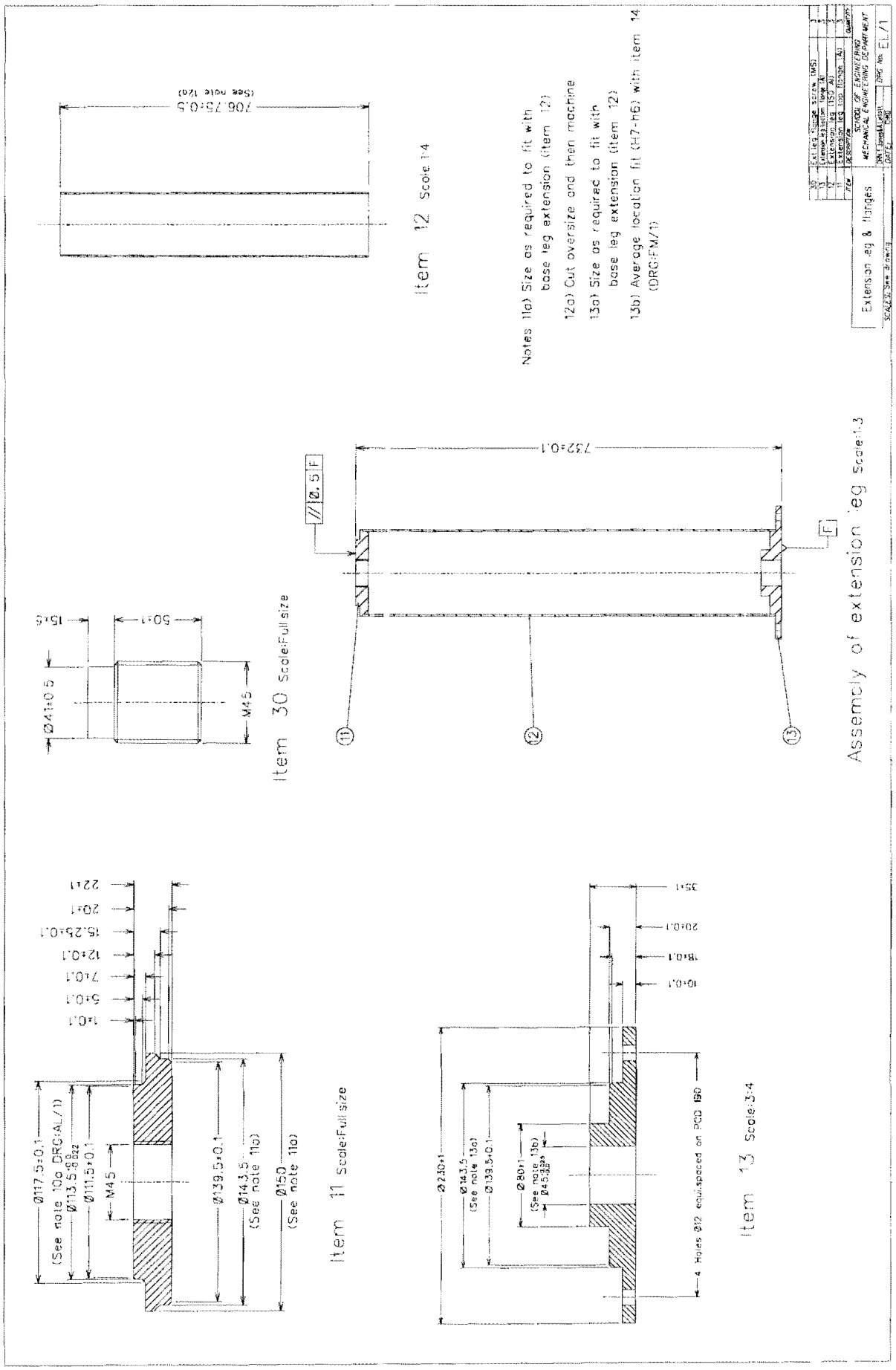


Figure A.10 Extension leg.

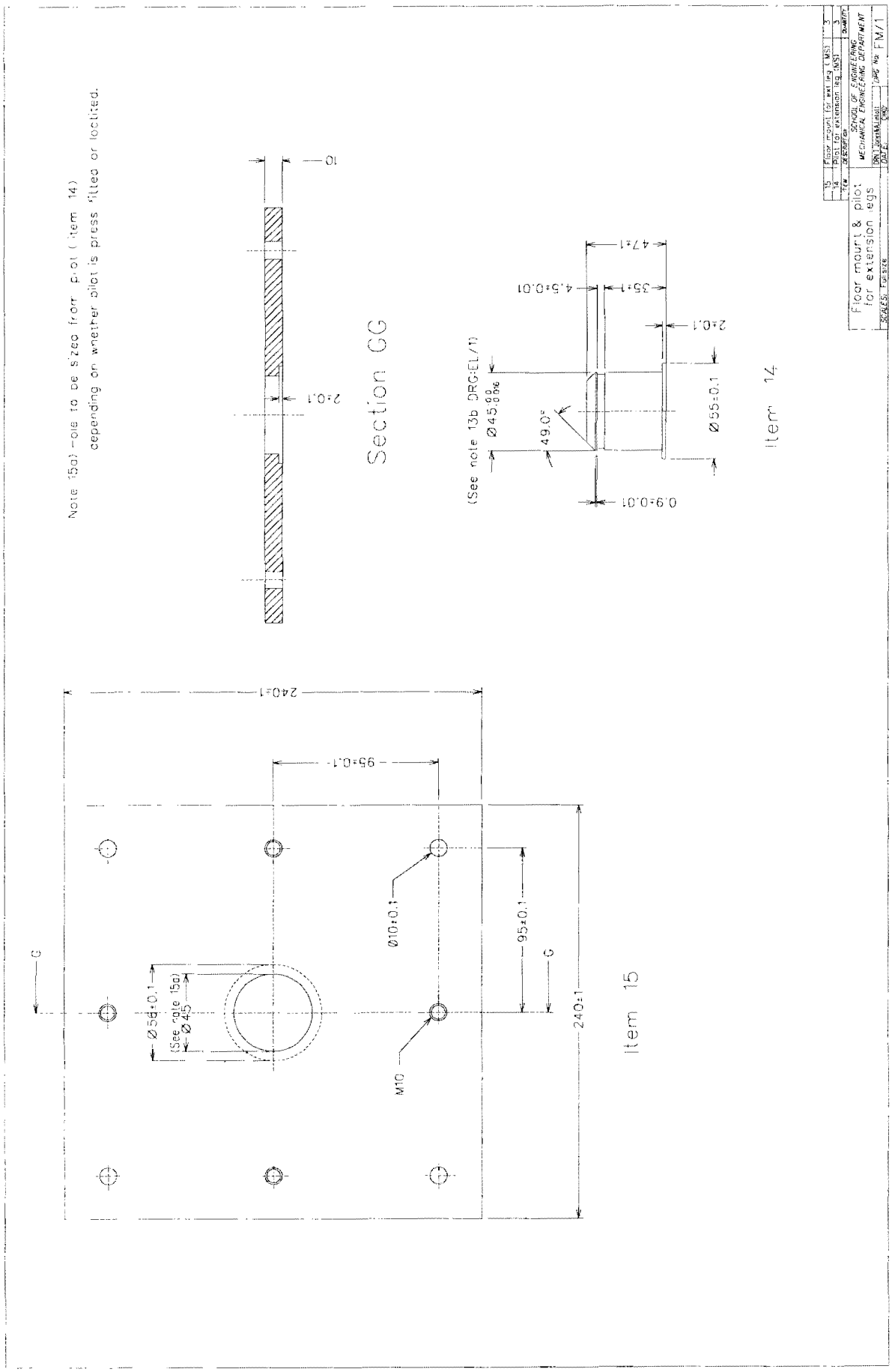


Figure A.11 Floor mount for support leg.

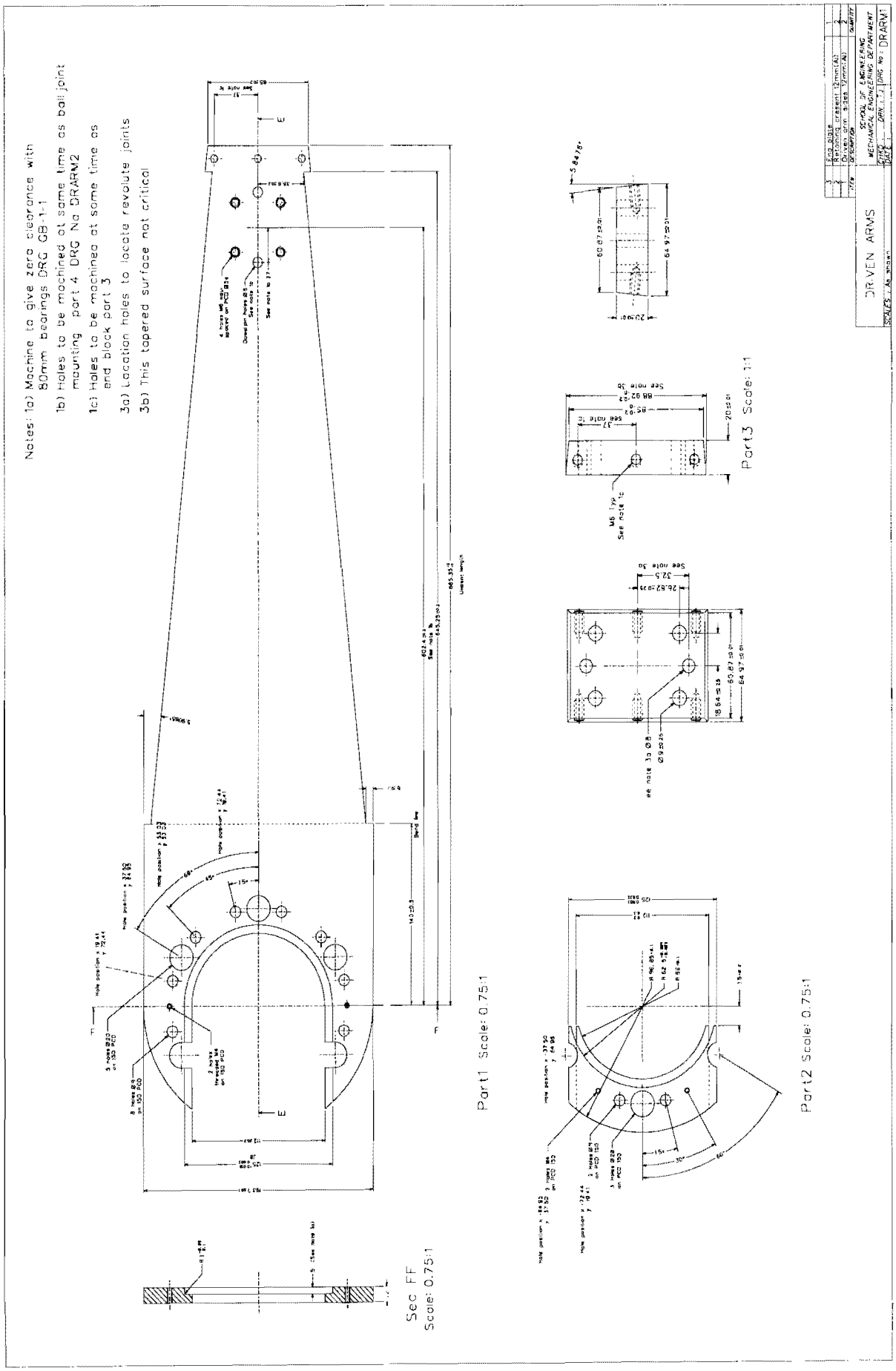


Figure A.12 Lower arm.

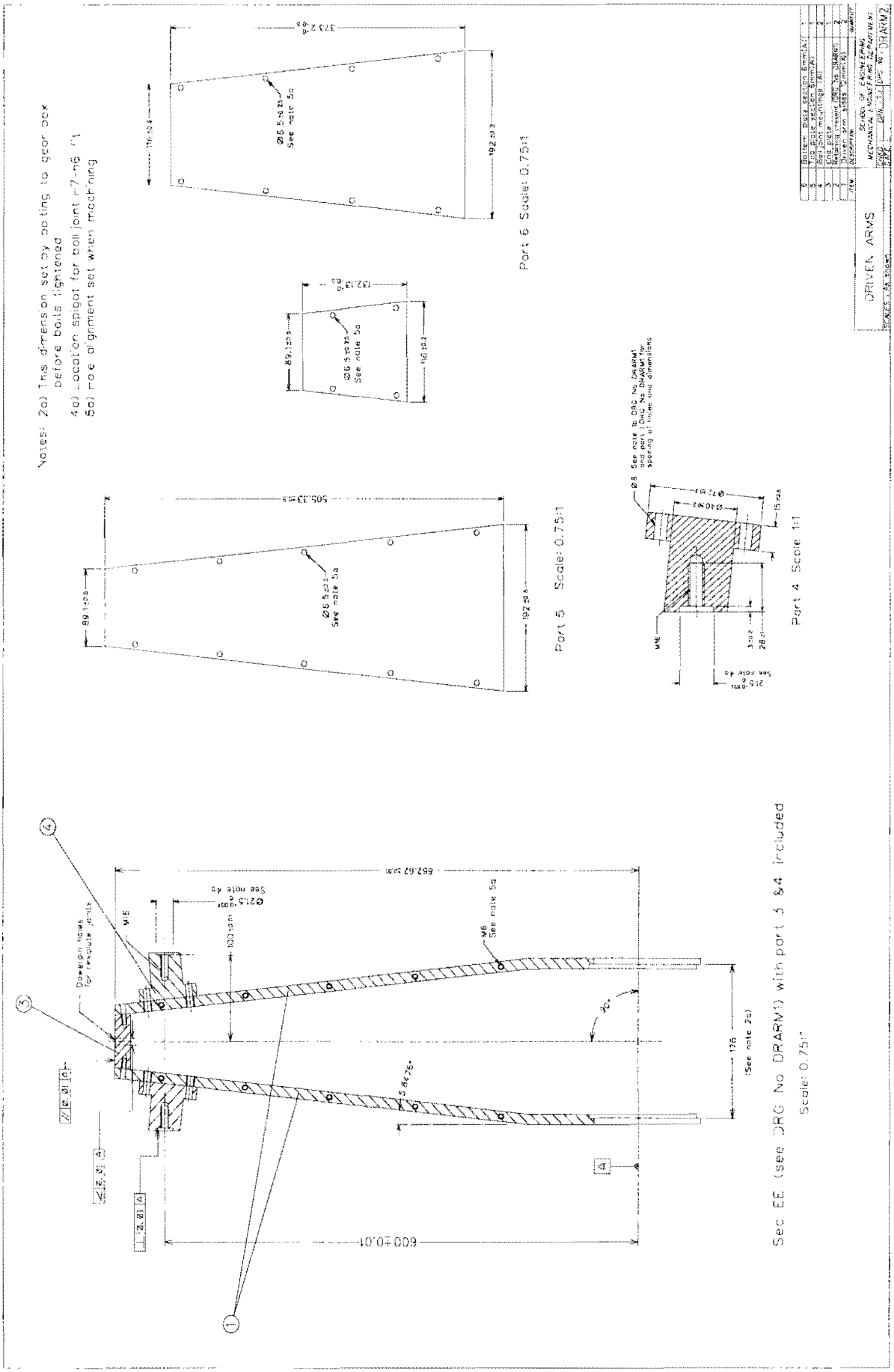


Figure A.13 Lower arm.

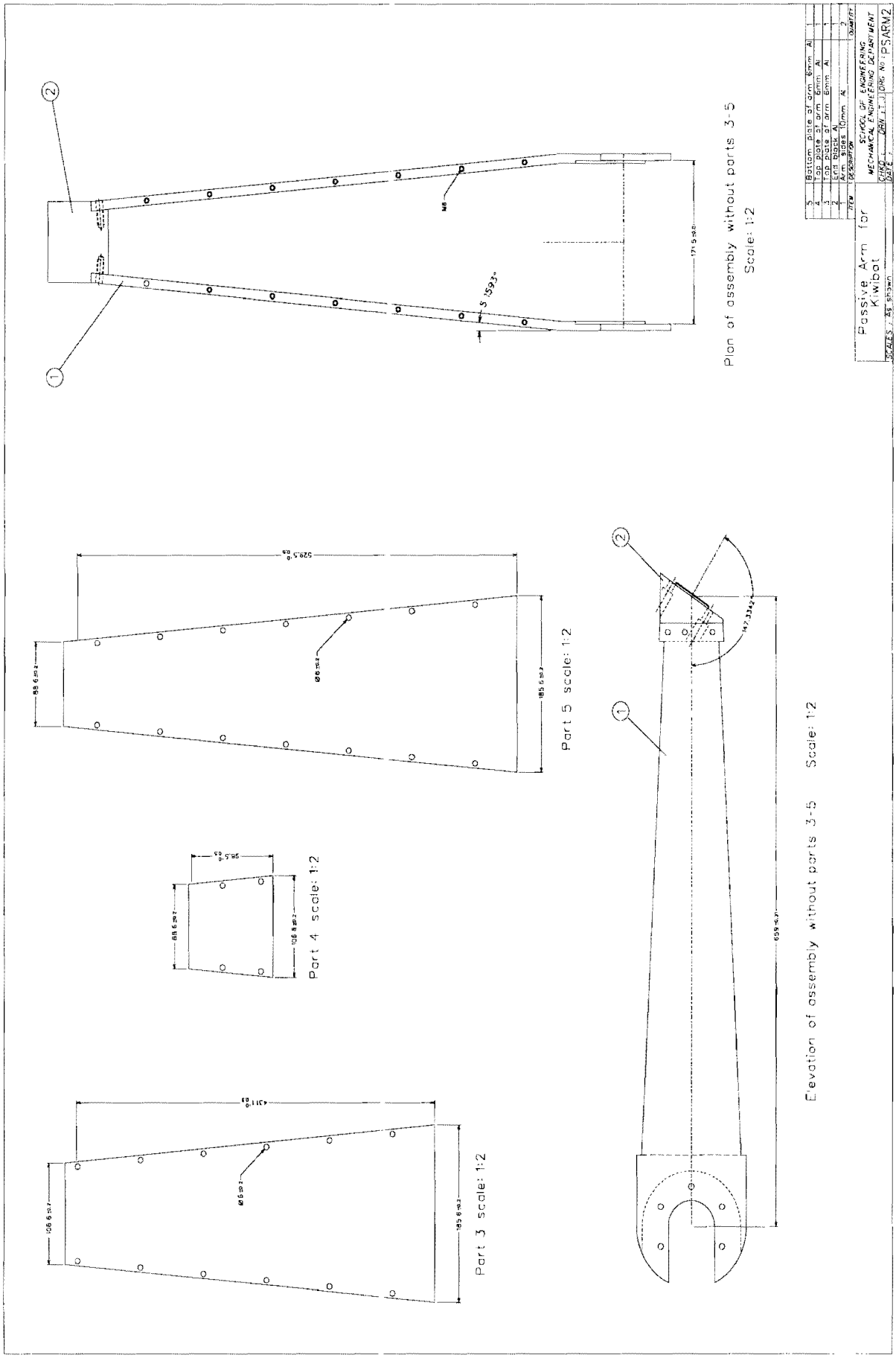


Figure A.14 Upper arm.

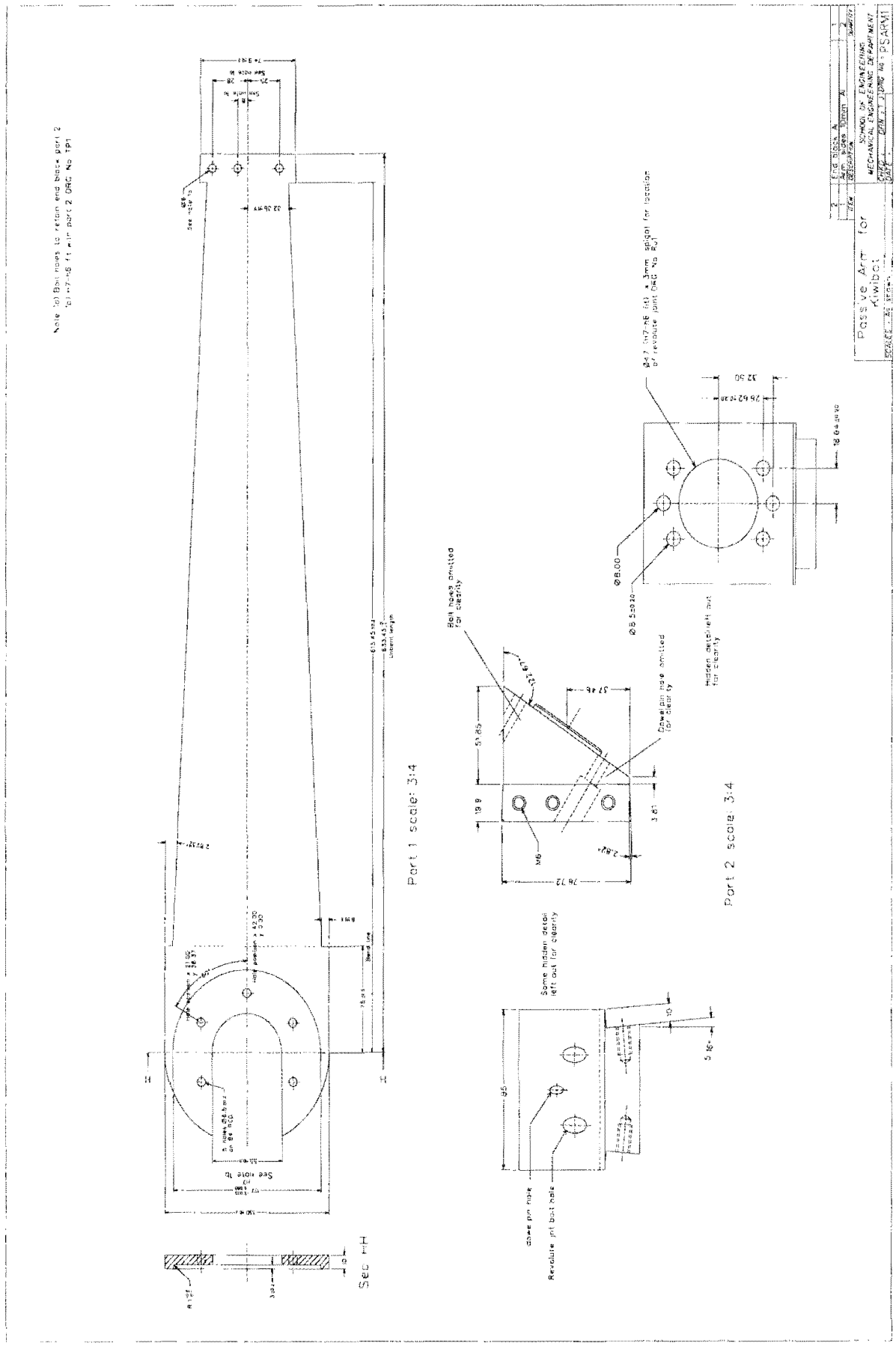


Figure A.15 Upper arm.

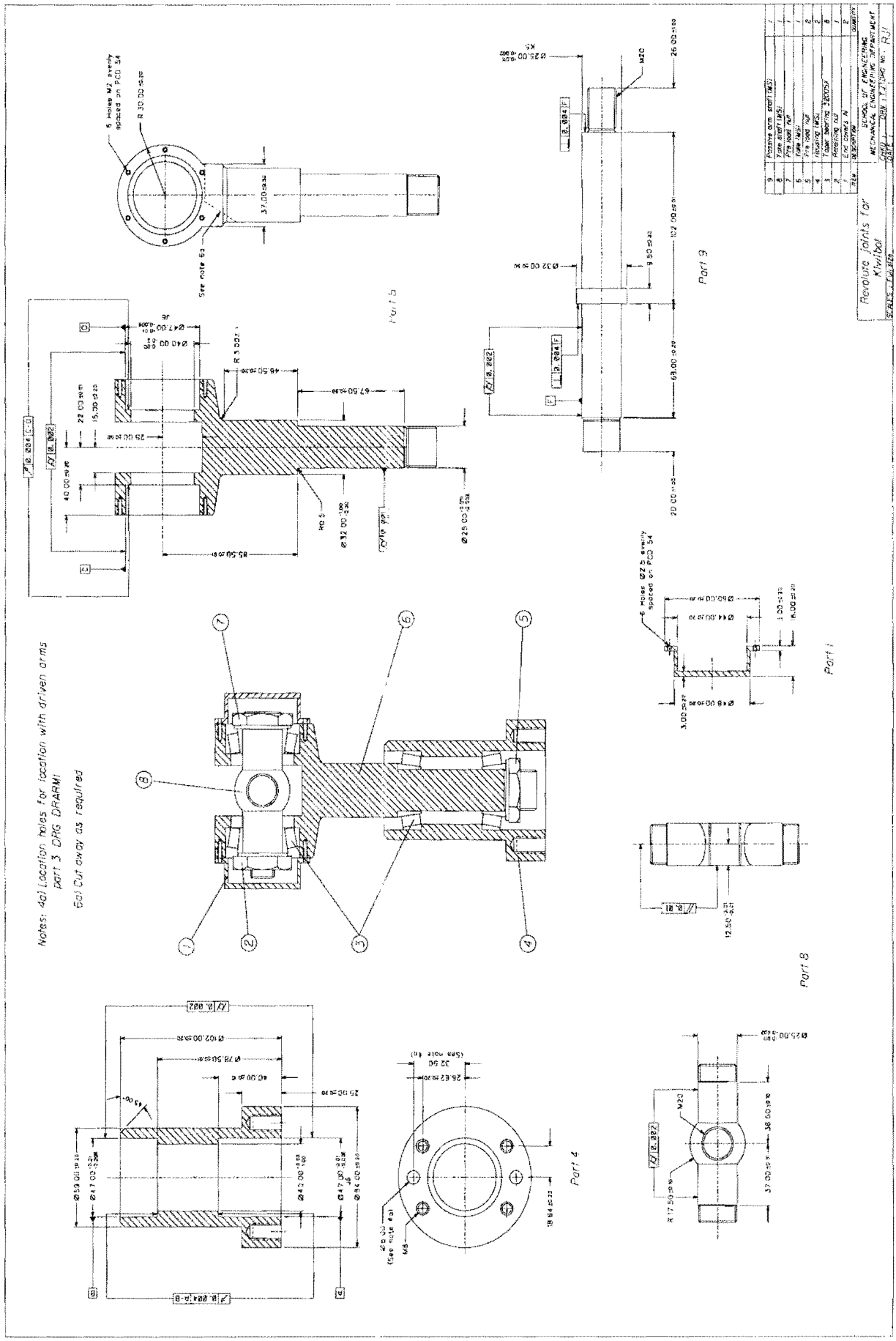


Figure A.16 Revolute joints.

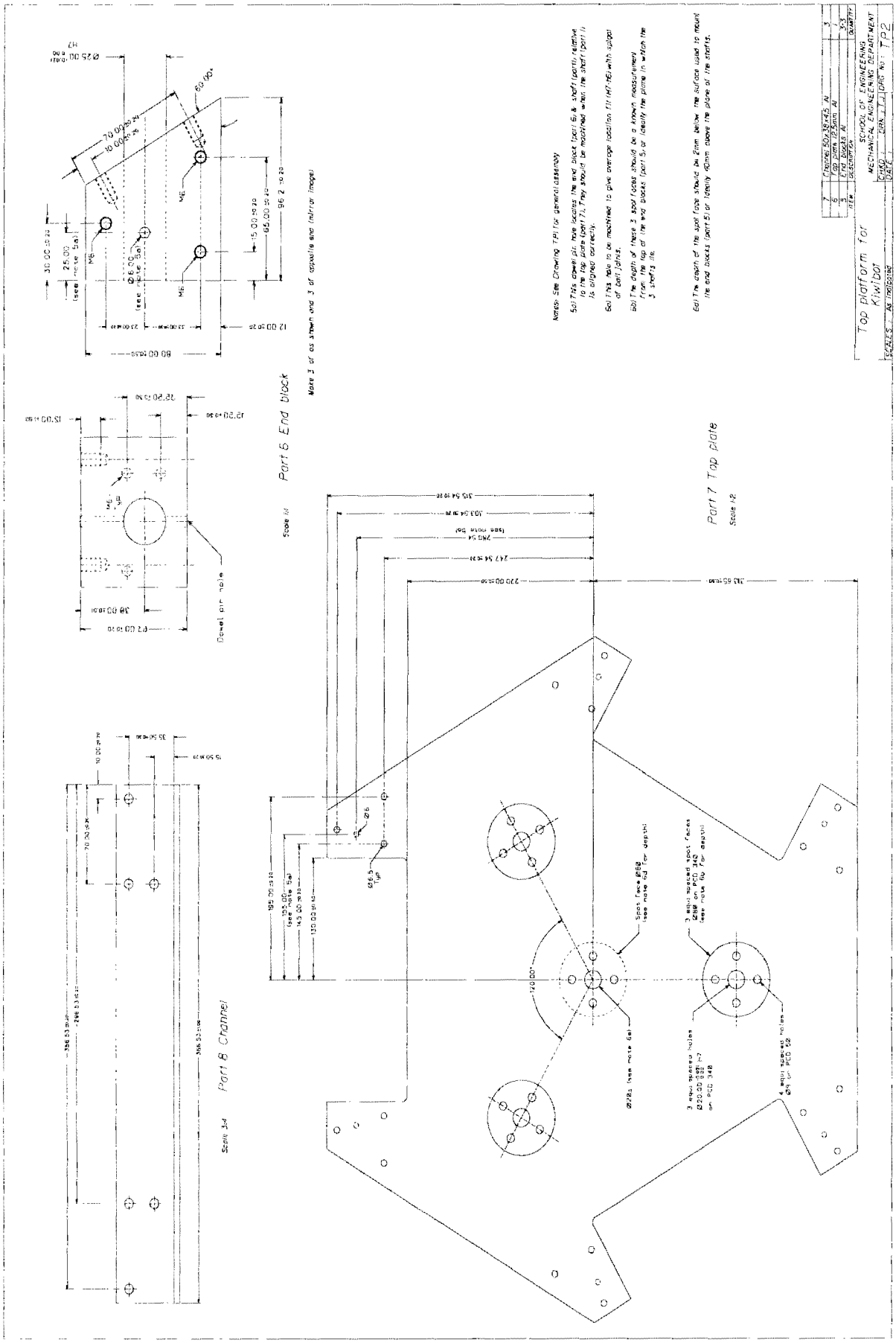


Figure A.17 Platform.

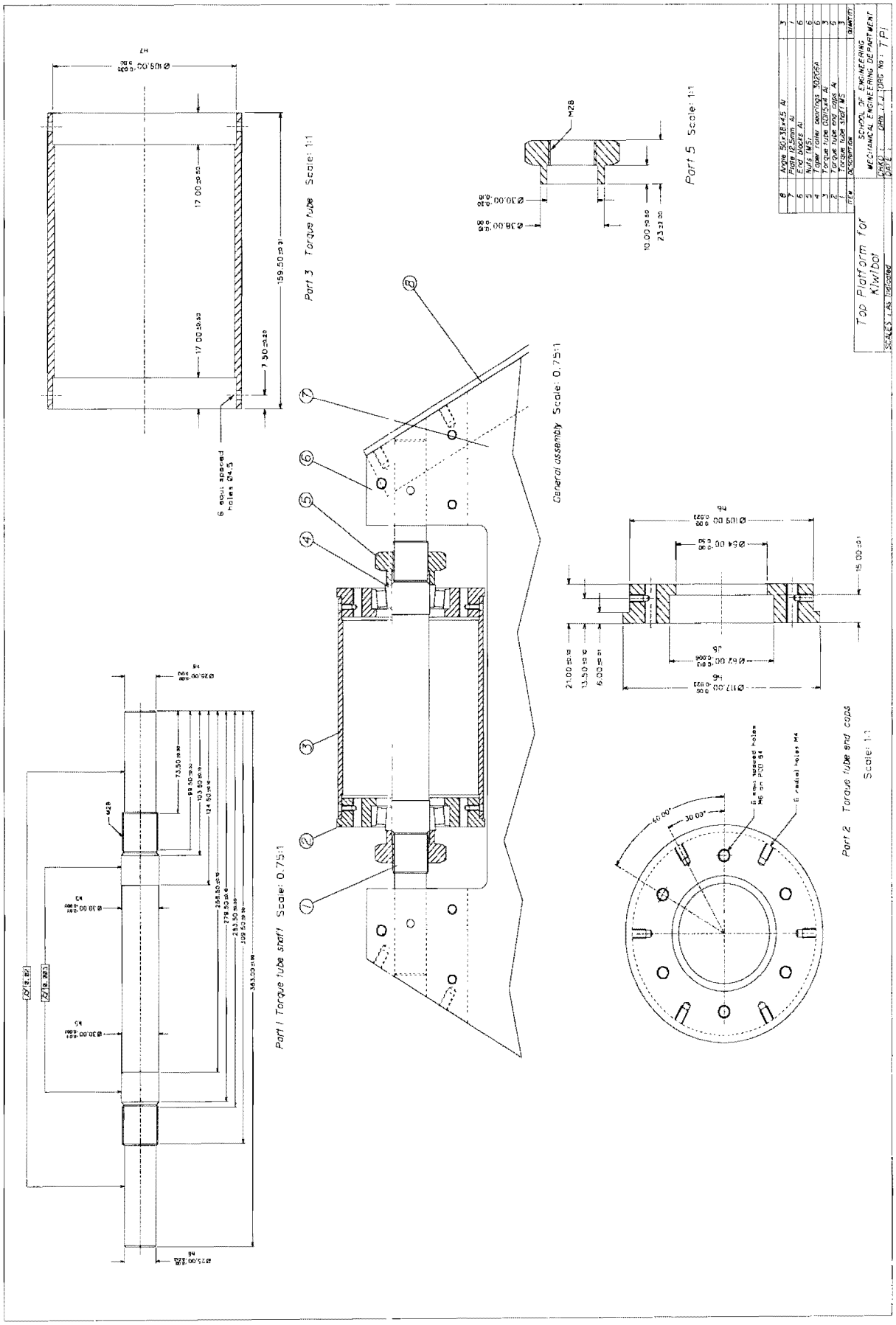


Figure A.18 Platform.

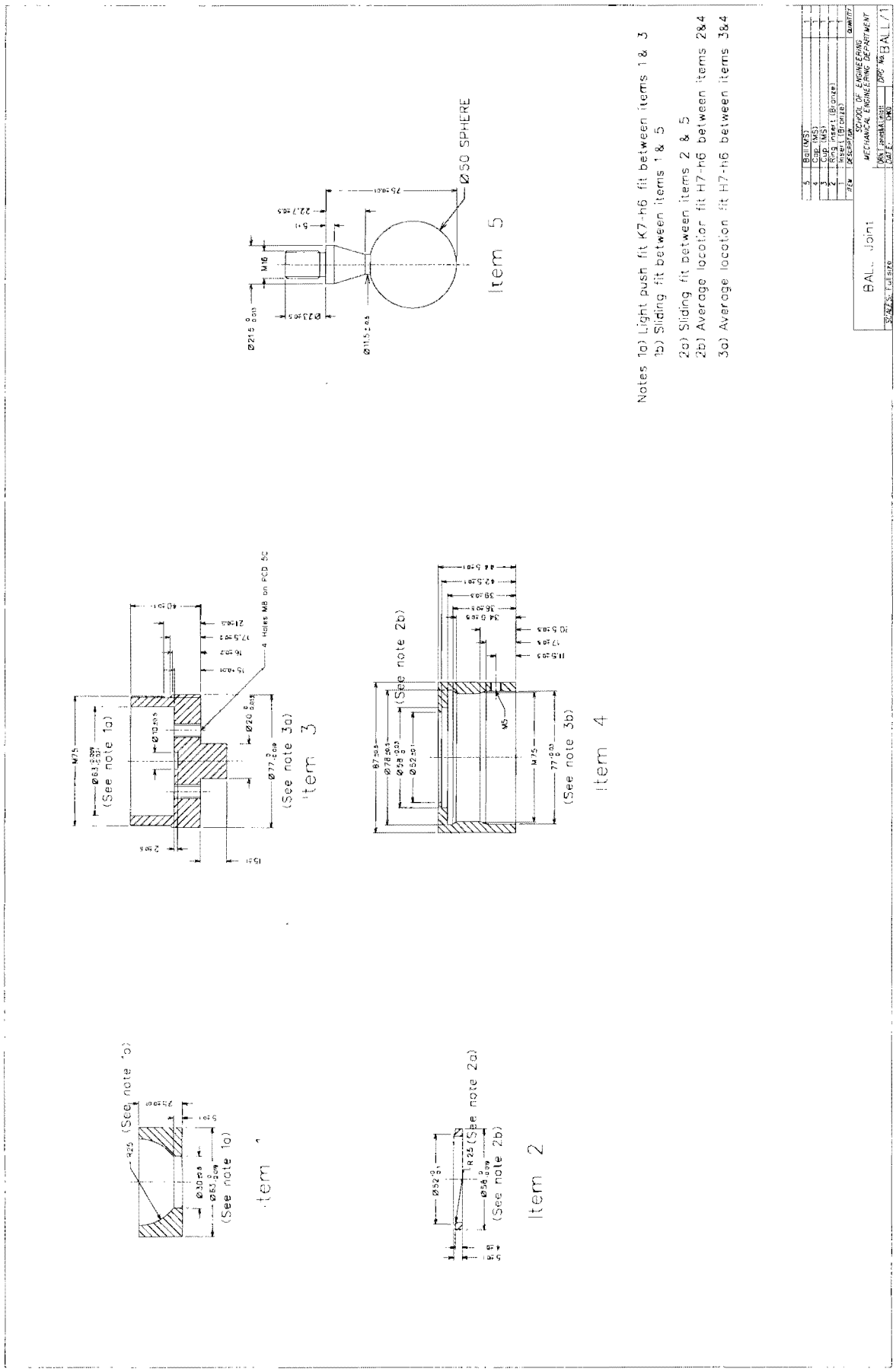


Figure A.19 Ball joint.

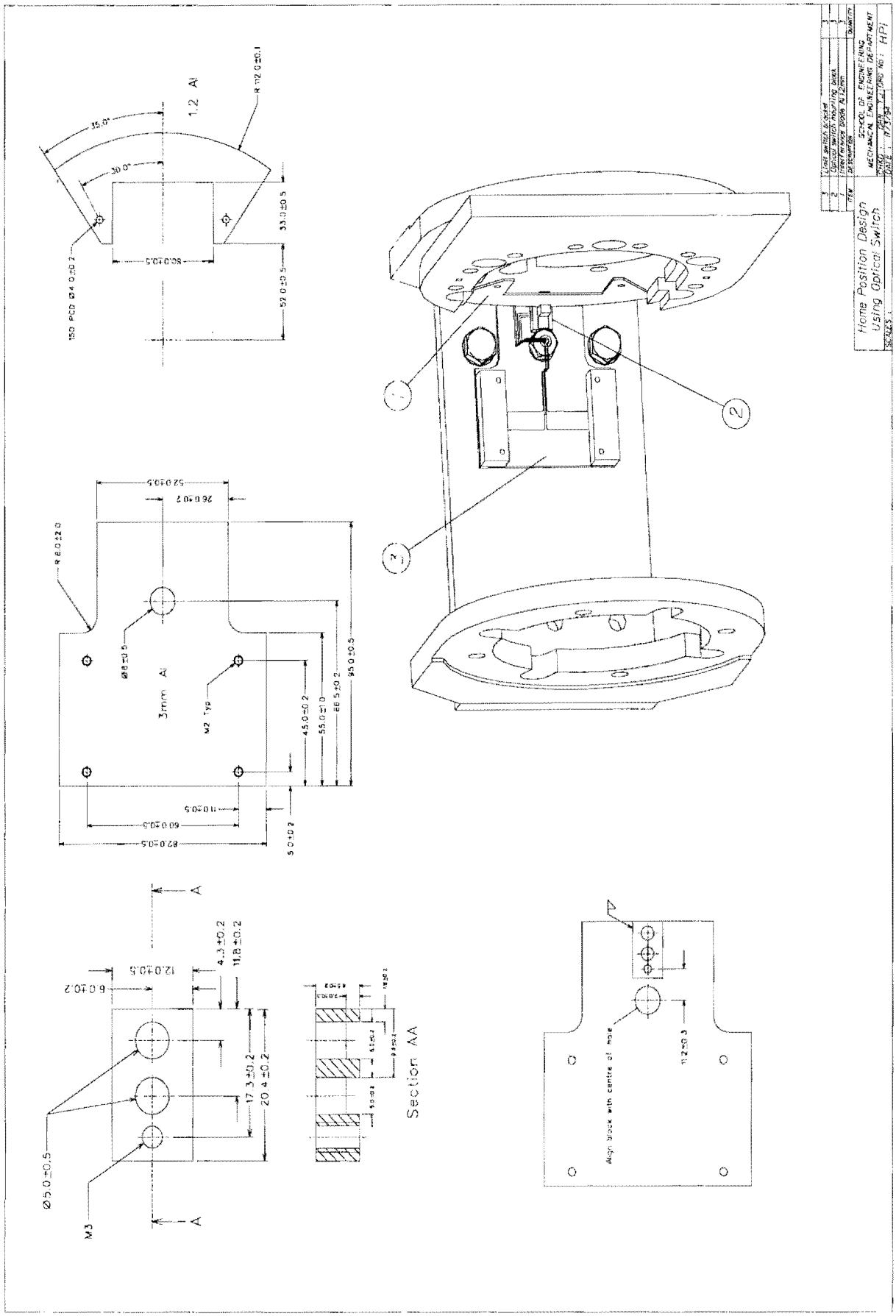


Figure A.20 Home and limit switch mounting bracket.

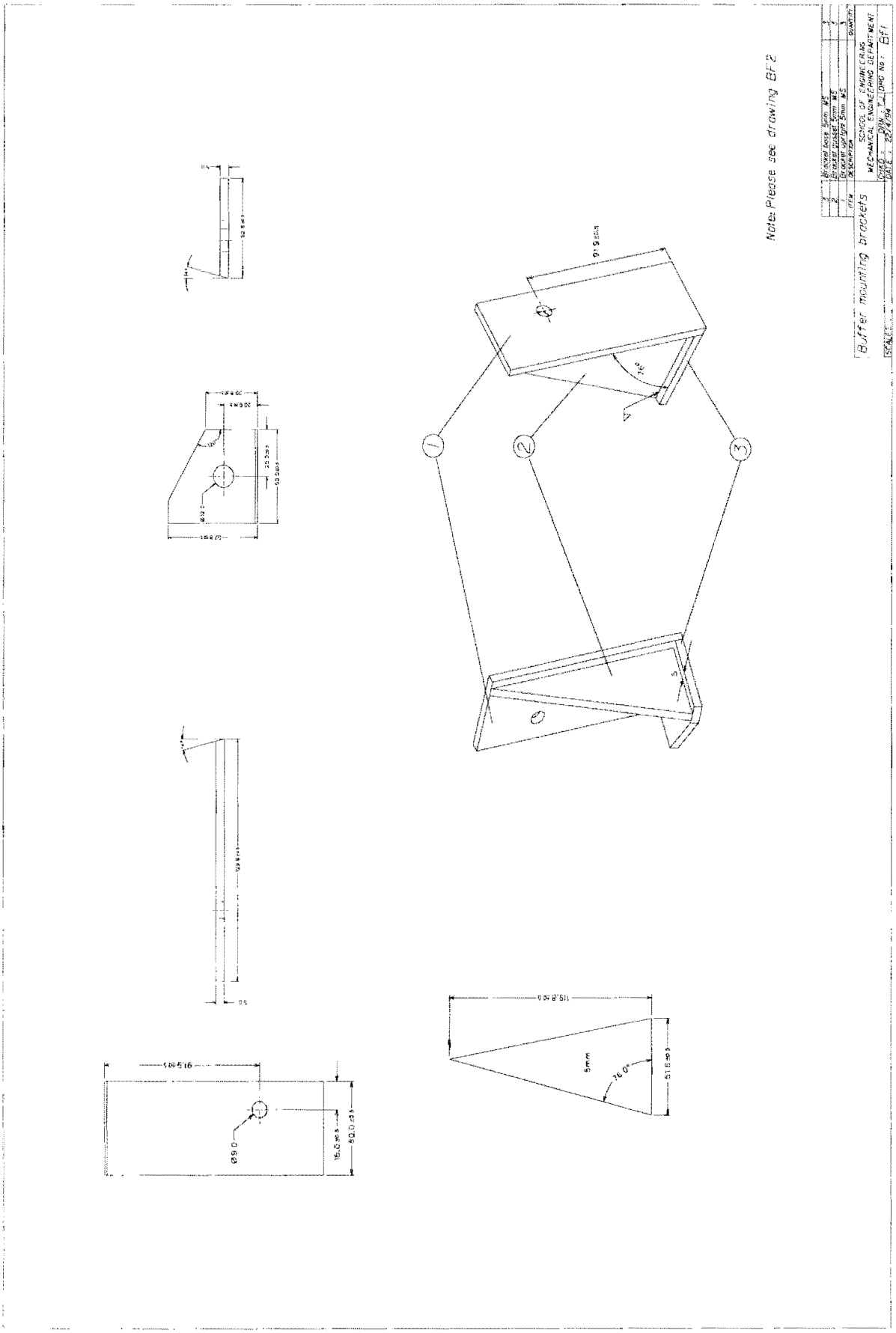


Figure A.21 Buffer mounting brackets.

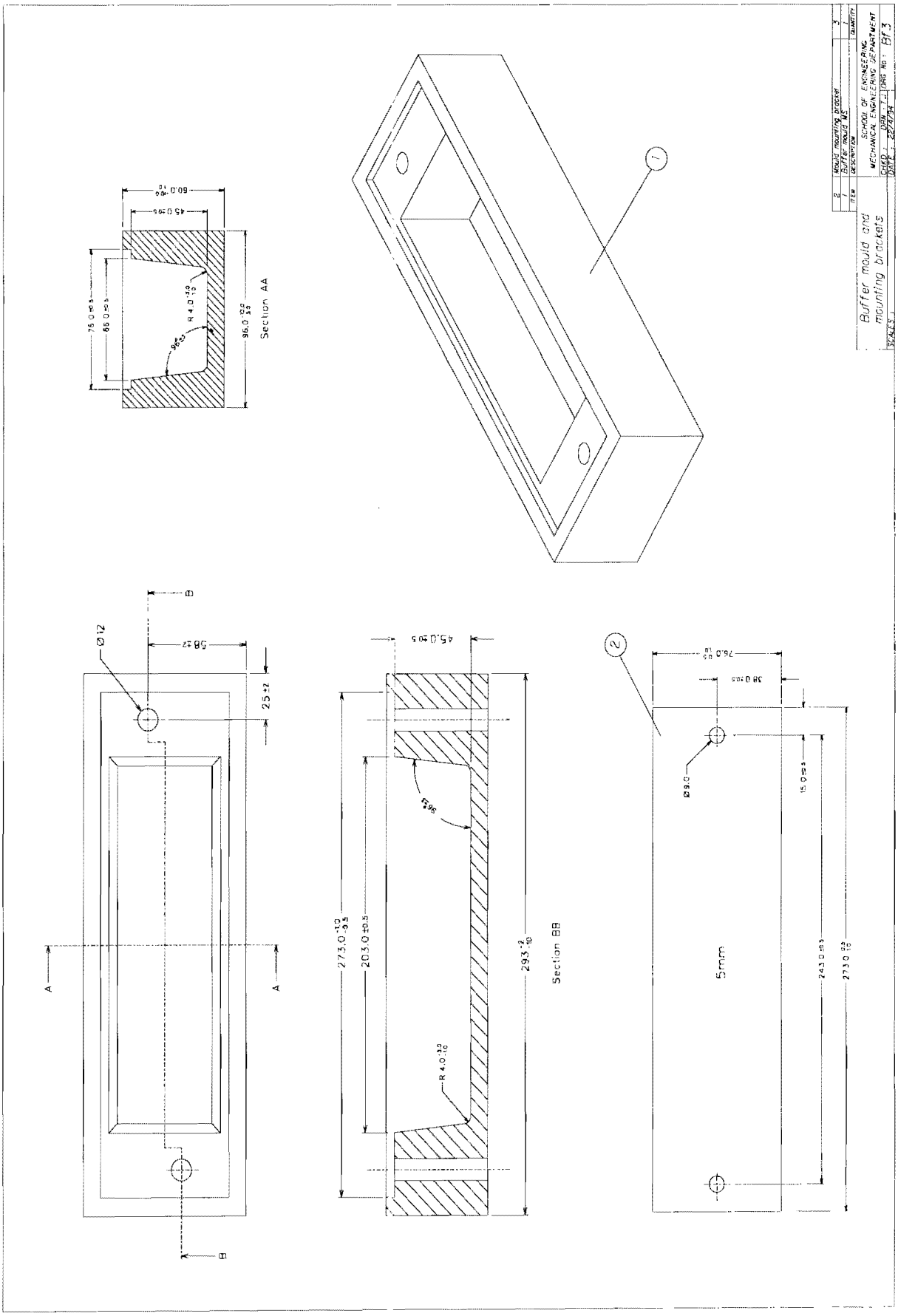


Figure A.22 Buffer mould.

Appendix B

Kinematics

B.1 Constants for the closure equations

Listed below are the constants for the closure equations given in section 5.2.1.

$$
\begin{aligned}
d_1 = {} & 2.a_{4x}.l_4.P_{4x} \\
& - 2.a_{4x}.l_4.P_{5x} \\
& + 2.a_{4y}.l_4.P_{4y} \\
& - 2.a_{4y}.l_4.P_{5y} \\
& + 2.a_{4z}.l_4.P_{4z} \\
& - 2.a_{4z}.l_4.P_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_1 = {} & 2.a_{5x}.l_5.P_{5x} \\
& - 2.a_{5x}.l_5.P_{6x} \\
& + 2.a_{5y}.l_5.P_{5y} \\
& - 2.a_{5y}.l_5.P_{6y} \\
& + 2.a_{5z}.l_5.P_{5z} \\
& - 2.a_{5z}.l_5.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_1 = {} & - 2.a_{6x}.l_6.P_{4x} \\
& + 2.a_{6x}.l_6.P_{6x} \\
& - 2.a_{6y}.l_6.P_{4y} \\
& + 2.a_{6y}.l_6.P_{6y} \\
& - 2.a_{6z}.l_6.P_{4z} \\
& + 2.a_{6z}.l_6.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_2 = {} & - 2.a_{5x}.l_5.P_{4x} \\
& + 2.a_{5x}.l_5.P_{5x} \\
& - 2.a_{5y}.l_5.P_{4y} \\
& + 2.a_{5y}.l_5.P_{5y} \\
& - 2.a_{5z}.l_5.P_{4z} \\
& + 2.a_{5z}.l_5.P_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_2 = {} & - 2.a_{6x}.l_6.P_{5x} \\
& + 2.a_{6x}.l_6.P_{6x} \\
& - 2.a_{6y}.l_6.P_{5y} \\
& + 2.a_{6y}.l_6.P_{6y} \\
& - 2.a_{6z}.l_6.P_{5z} \\
& + 2.a_{6z}.l_6.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_2 = {} & 2.a_{4x}.l_4.P_{4x} \\
& - 2.a_{4x}.l_4.P_{6x} \\
& + 2.a_{4y}.l_4.P_{4y} \\
& - 2.a_{4y}.l_4.P_{6y} \\
& + 2.a_{4z}.l_4.P_{4z} \\
& - 2.a_{4z}.l_4.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_3 = {} & 2.l_4.n_{4x}.P_{4x} \\
& - 2.l_4.n_{4x}.P_{5x} \\
& + 2.l_4.n_{4y}.P_{4y} \\
& - 2.l_4.n_{4y}.P_{5y} \\
& + 2.l_4.n_{4z}.P_{4z} \\
& - 2.l_4.n_{4z}.P_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_3 = {} & 2.l_5.n_{5x}.P_{5x} \\
& - 2.l_5.n_{5x}.P_{6x} \\
& + 2.l_5.n_{5y}.P_{5y} \\
& - 2.l_5.n_{5y}.P_{6y} \\
& + 2.l_5.n_{5z}.P_{5z} \\
& - 2.l_5.n_{5z}.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_3 = {} & - 2.l_6.n_{6x}.P_{4x} \\
& + 2.l_6.n_{6x}.P_{6x} \\
& - 2.l_6.n_{6y}.P_{4y} \\
& + 2.l_6.n_{6y}.P_{6y} \\
& - 2.l_6.n_{6z}.P_{4z} \\
& + 2.l_6.n_{6z}.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_4 = {} & - 2.l_5.n_{5x}.P_{4x} \\
& + 2.l_5.n_{5x}.P_{5x} \\
& - 2.l_5.n_{5y}.P_{4y} \\
& + 2.l_5.n_{5y}.P_{5y} \\
& - 2.l_5.n_{5z}.P_{4z} \\
& + 2.l_5.n_{5z}.P_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_4 = {} & - 2.l_6.n_{6x}.P_{5x} \\
& + 2.l_6.n_{6x}.P_{6x} \\
& - 2.l_6.n_{6y}.P_{5y} \\
& + 2.l_6.n_{6y}.P_{6y} \\
& - 2.l_6.n_{6z}.P_{5z} \\
& + 2.l_6.n_{6z}.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_4 = {} & + 2.l_4.n_{4x}.P_{4x} \\
& - 2.l_4.n_{4x}.P_{6x} \\
& + 2.l_4.n_{4y}.P_{4y} \\
& - 2.l_4.n_{4y}.P_{6y} \\
& + 2.l_4.n_{4z}.P_{4z} \\
& - 2.l_4.n_{4z}.P_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_5 = {} & - 2.a_{4x}.a_{5x}.l_4.l_5 \\
& - 2.a_{4y}.a_{5y}.l_4.l_5 \\
& - 2.a_{4z}.a_{5z}.l_4.l_5
\end{aligned}
$$

$$
\begin{aligned}
h_5 = {} & - 2.a_{5x}.a_{6x}.l_5.l_6 \\
& - 2.a_{5y}.a_{6y}.l_5.l_6 \\
& - 2.a_{5z}.a_{6z}.l_5.l_6
\end{aligned}
$$

$$
\begin{aligned}
l_5 = {} & - 2.a_{4x}.a_{6x}.l_4.l_6 \\
& - 2.a_{4y}.a_{6y}.l_4.l_6 \\
& - 2.a_{4z}.a_{6z}.l_4.l_6
\end{aligned}
$$

$$
\begin{aligned}
d_6 = {} & - 2.a_{5x}.l_4.l_5.n_{4x} \\
& - 2.a_{5y}.l_4.l_5.n_{4y} \\
& - 2.a_{5z}.l_4.l_5.n_{4z}
\end{aligned}
$$

$$
\begin{aligned}
h_6 = {} & - 2.a_{6x}.l_5.l_6.n_{5x} \\
& - 2.a_{6y}.l_5.l_6.n_{5y} \\
& - 2.a_{6z}.l_5.l_6.n_{5z}
\end{aligned}
$$

$$
\begin{aligned}
l_6 = {} & - 2.a_{4x}.l_4.l_6.n_{6x} \\
& - 2.a_{4y}.l_4.l_6.n_{6y} \\
& - 2.a_{4z}.l_4.l_6.n_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_7 = {} & - 2.a_{4x}.l_4.l_5.n_{5x} \\
& - 2.a_{4y}.l_4.l_5.n_{5y} \\
& - 2.a_{4z}.l_4.l_5.n_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_7 = {} & - 2.a_{5x}.l_5.l_6.n_{6x} \\
& - 2.a_{5y}.l_5.l_6.n_{6y} \\
& - 2.a_{5z}.l_5.l_6.n_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_7 = {} & - 2.a_{6x}.l_4.l_6.n_{4x} \\
& - 2.a_{6y}.l_4.l_6.n_{4y} \\
& - 2.a_{6z}.l_4.l_6.n_{4z}
\end{aligned}
$$

$$
\begin{aligned}
d_8 = {} & - 2.l_4.l_5.n_{4x}.n_{5x} \\
& - 2.l_4.l_5.n_{4y}.n_{5y} \\
& - 2.l_4.l_5.n_{4z}.n_{5z}
\end{aligned}
$$

$$
\begin{aligned}
h_8 = {} & - 2.l_5.l_6.n_{5x}.n_{6x} \\
& - 2.l_5.l_6.n_{5y}.n_{6y} \\
& - 2.l_5.l_6.n_{5z}.n_{6z}
\end{aligned}
$$

$$
\begin{aligned}
l_8 = {} & - 2.l_4.l_6.n_{4x}.n_{6x} \\
& - 2.l_4.l_6.n_{4y}.n_{6y} \\
& - 2.l_4.l_6.n_{4z}.n_{6z}
\end{aligned}
$$

$$
\begin{aligned}
d_9 = {} & -e^2 \\
& + P_{4x}^2 \\
& - 2.P_{4x}.P_{5x} \\
& + P_{5x}^2 \\
& + P_{4y}^2 \\
& - 2.P_{4y}.P_{5y} \\
& + P_{5y}^2 \\
& + P_{4z}^2 \\
& - 2.P_{4z}.P_{5z} \\
& + P_{5z}^2 \\
& + l_4^2 \\
& + l_5^2
\end{aligned}
$$

$$
\begin{aligned}
h_9 = {} & -f^2 \\
& + P_{5x}^2 \\
& - 2.P_{5x}.P_{6x} \\
& + P_{6x}^2 \\
& + P_{5y}^2 \\
& - 2.P_{5y}.P_{6y} \\
& + P_{6y}^2 \\
& + P_{5z}^2 \\
& - 2.P_{5z}.P_{6z} \\
& + P_{6z}^2 \\
& + l_5^2 \\
& + l_6^2
\end{aligned}
$$

$$
\begin{aligned}
l_9 = {} & -g^2 \\
& + P_{4x}^2 \\
& - 2.P_{4x}.P_{6x} \\
& + P_{6x}^2 \\
& + P_{4y}^2 \\
& - 2.P_{4y}.P_{6y} \\
& + P_{6y}^2 \\
& + P_{4z}^2 \\
& - 2.P_{4z}.P_{6z} \\
& + P_{6z}^2 \\
& + l_4^2 \\
& + l_6^2
\end{aligned}
$$

B.2 The intersection of a circle and a plane using vector notation

The following section solves for the two points of intersection between a plane and circle at any arbitrary position or orientation.

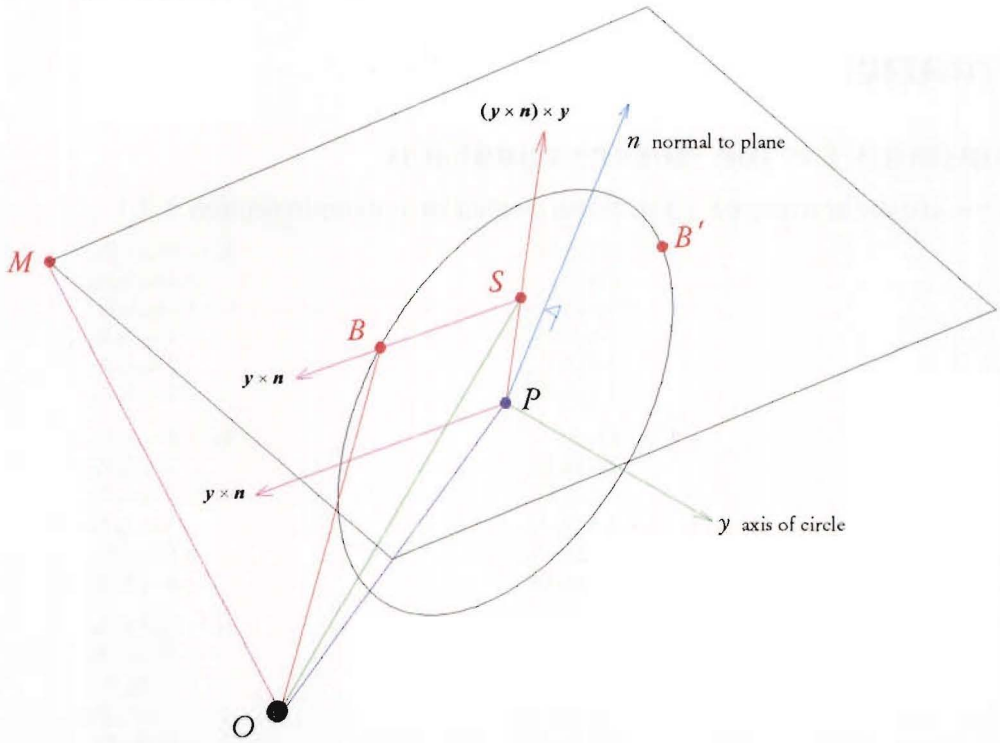


Figure B.1 Vectors defining the orientation and position of the circle and plane (note, points lying in the plane are shown in red).

The following parameters relative to some reference point O must be known to find the two intersection points B and B' of the circle and plane.

$\boldsymbol{n}$ is the normal to the plane thus defining the orientation of the plane.
P is the point at the centre of the circle.
$\boldsymbol{M}$ is some point in the plane.
$\boldsymbol{y}$ is a vector normal to the circle thus defining the orientation of the circle.
l is the radius of the circle.

The point S which is the point midway between the intersection points B and B' can now be found. The point S is the intersection of the vector $(\boldsymbol{y} \times \boldsymbol{n}) \times \boldsymbol{y}$ through P and the plane. The equation of the plane can be written as

$$\boldsymbol{n}^T \boldsymbol{S} - \boldsymbol{n}^T \boldsymbol{M} = 0 . \tag{B.1}$$

The position vector $\boldsymbol{S}$ can be written in parametric form as

$$\boldsymbol{S} = \boldsymbol{P} + f \{(\boldsymbol{y} \times \boldsymbol{n}) \times \boldsymbol{y}\} \tag{B.2}$$

where f is a scalar quantity obtained by combining equations (B.1) and (B.2) and is given by

$$f = \frac{\boldsymbol{n}^T \boldsymbol{M} - \boldsymbol{n}^T \boldsymbol{P}}{\boldsymbol{n}^T \{(\boldsymbol{y} \times \boldsymbol{n}) \times \boldsymbol{y}\}} .$$

The vector $\overline{SB}$ can be written as

$$\overline{SB} = t(\boldsymbol{y} \times \boldsymbol{n})$$

where t is some unknown scalar quantity.

The position vector $\boldsymbol{B}$, the position of the point of intersection between the plane and circle is given by

$$\boldsymbol{B} = \boldsymbol{S} + \overline{SB}\,.$$

The distance from the point P (centre of the circle) to the points B and B' (points of intersection with the homokinetic plane) must be length l. Hence

$$l = \left|\boldsymbol{B} - \boldsymbol{P}\right|$$

or

$$l = \left|\boldsymbol{S} + t(\boldsymbol{y} \times \boldsymbol{n}) - \boldsymbol{P}\right|. \qquad \text{(B.3)}$$

Squaring both sides of equation (B.3) and rearranging gives a quadratic in the scalar quantity t. The solution for t is shown below. The intersection points $\boldsymbol{B}$ are then given by

$$\boldsymbol{B} = \boldsymbol{S} + t(\boldsymbol{y} \times \boldsymbol{n}).$$

t = -(2.n(3).P(2).y(1) - 2.n(2).P(3).y(1) - 2.n(3).S(2).y(1) + 2.n(2).S(3).y(1) - 2.n(3).P(1).y(2) + 2.n(1).P(3).y(2) + 2.n(3).S(1).y(2) - 2.n(1).S(3).y(2)
+ 2.n(2).P(1).y(3) - 2.n(1).P(2).y(3) - 2.n(2).S(1).y(3) + 2.n(1).S(2).y(3))/(n(2)^2.y(1)^2+ n(3)^2.y(1)^2- 2.n(1).n(2).y(1).y(2) + n(1)^2.y(2)^2+ n(3)^2.y(2)^2
- 2.n(1).n(3).y(1).y(3) - 2.n(2).n(3).y(2).y(3) + n(1)^2.y(3)^2+ n(2)^2.y(3)^2)

±(2.(l^2.n(2)^2.y(1)^2+ l^2.n(3)^2.y(1)^2- n(2)^2.P(1)^2.y(1)^2 - n(3)^2.P(1)^2.y(1)^2 - n(2)^2.P(2)^2.y(1)^2 - 2.n(2).n(3).P(2).P(3).y(1)^2 - n(3)^2.P(3)^2.y(1)^2
+ 2.n(2)^2.P(1).S(1).y(1)^2 + 2.n(3)^2.P(1).S(1).y(1)^2 - n(2)^2.S(1)^2.y(1)^2 - n(3)^2.S(1)^2.y(1)^2 + 2.n(2)^2.P(2).S(2).y(1)^2 + 2.n(2).n(3).P(3).S(2).y(1)^2
- n(2)^2.S(2)^2.y(1)^2 + 2.n(2).n(3).P(2).S(3).y(1)^2 + 2.n(3)^2.P(3).S(3).y(1)^2 - 2.n(2).n(3).S(2).S(3).y(1)^2 - n(3)^2.S(3)^2.y(1)^2 - 2.l^2.n(1).n(2).y(1).y(2)
+ 2.n(1).n(2).P(1)^2.y(1).y(2) - 2.n(3)^2.P(1).P(2).y(1).y(2) + 2.n(1).n(2).P(2)^2.y(1).y(2) + 2.n(2).n(3).P(1).P(3).y(1).y(2) + 2.n(1).n(3).P(2).P(3).y(1).y(2)
- 4.n(1).n(2).P(1).S(1).y(1).y(2) + 2.n(3)^2.P(2).S(1).y(1).y(2) - 2.n(2).n(3).P(3).S(1).y(1).y(2) + 2.n(1).n(2).S(1)^2.y(1).y(2) + 2.n(3)^2.P(1).S(2).y(1).y(2)
- 4.n(1).n(2).P(2).S(2).y(1).y(2) - 2.n(1).n(3).P(3).S(2).y(1).y(2) - 2.n(3)^2.S(1).S(2).y(1).y(2) + 2.n(1).n(2).S(2)^2.y(1).y(2)
- 2.n(2).n(3).P(1).S(3).y(1).y(2) - 2.n(1).n(3).P(2).S(3).y(1).y(2) + 2.n(2).n(3).S(1).S(3).y(1).y(2) + 2.n(1).n(3).S(2).S(3).y(1).y(2) + l^2.n(1)^2.y(2)^2
+ l^2.n(3)^2.y(2)^2 - n(1)^2.P(1)^2.y(2)^2 - n(1)^2.P(2)^2.y(2)^2 - n(3)^2.P(2)^2.y(2)^2 - 2.n(1).n(3).P(1).P(3).y(2)^2 - n(3)^2.P(3)^2.y(2)^2 + 2.n(1)^2.P(1).S(1).y(2)^2
+ 2.n(1).n(3).P(3).S(1).y(2)^2 - n(1)^2.S(1)^2.y(2)^2 + 2.n(1)^2.P(2).S(2).y(2)^2 + 2.n(3)^2.P(2).S(2).y(2)^2 - n(1)^2.S(2)^2.y(2)^2 - n(3)^2.S(2)^2.y(2)^2
+ 2.n(1).n(3).P(1).S(3).y(2)^2 + 2.n(3)^2.P(3).S(3).y(2)^2 - 2.n(1).n(3).S(1).S(3).y(2)^2 - n(3)^2.S(3)^2.y(2)^2 - 2.l^2.n(1).n(3).y(1).y(3)
+ 2.n(1).n(3).P(1)^2.y(1).y(3) + 2.n(2).n(3).P(1).P(2).y(1).y(3) - 2.n(2)^2.P(1).P(3).y(1).y(3) + 2.n(1).n(2).P(2).P(3).y(1).y(3) + 2.n(1).n(3).P(3)^2.y(1).y(3)
- 4.n(1).n(3).P(1).S(1).y(1).y(3) - 2.n(2).n(3).P(2).S(1).y(1).y(3) + 2.n(2)^2.P(3).S(1).y(1).y(3) + 2.n(1).n(3).S(1)^2.y(1).y(3)
- 2.n(2).n(3).P(1).S(2).y(1).y(3) - 2.n(1).n(2).P(3).S(2).y(1).y(3) + 2.n(2).n(3).S(1).S(2).y(1).y(3) + 2.n(2)^2.P(1).S(3).y(1).y(3)
- 2.n(1).n(2).P(2).S(3).y(1).y(3) - 4.n(1).n(3).P(3).S(3).y(1).y(3) - 2.n(2)^2.S(1).S(3).y(1).y(3) + 2.n(1).n(2).S(2).S(3).y(1).y(3) + 2.n(1).n(3).S(3)^2.y(1).y(3)
- 2.l^2.n(2).n(3).y(2).y(3) + 2.n(1).n(3).P(1).P(2).y(2).y(3) + 2.n(2).n(3).P(2)^2.y(2).y(3) + 2.n(1).n(2).P(1).P(3).y(2).y(3) - 2.n(1)^2.P(2).P(3).y(2).y(3)
+ 2.n(2).n(3).P(3)^2.y(2).y(3) - 2.n(1).n(3).P(2).S(1).y(2).y(3) - 2.n(1).n(2).P(3).S(1).y(2).y(3) - 2.n(1).n(3).P(1).S(2).y(2).y(3)
- 4.n(2).n(3).P(2).S(2).y(2).y(3) + 2.n(1)^2.P(3).S(2).y(2).y(3) + 2.n(1).n(3).S(1).S(2).y(2).y(3) + 2.n(2).n(3).S(2)^2.y(2).y(3)
- 2.n(1).n(2).P(1).S(3).y(2).y(3) + 2.n(1)^2.P(2).S(3).y(2).y(3) - 4.n(2).n(3).P(3).S(3).y(2).y(3) + 2.n(1).n(2).S(1).S(3).y(2).y(3)
- 2.n(1)^2.S(2).S(3).y(2).y(3) + 2.n(2).n(3).S(3)^2.y(2).y(3) + l^2.n(1)^2.y(3)^2 + l^2.n(2)^2.y(3)^2 - n(1)^2.P(1)^2.y(3)^2 - 2.n(1).n(2).P(1).P(2).y(3)^2
- n(2)^2.P(2)^2.y(3)^2 - n(1)^2.P(3)^2.y(3)^2 - n(2)^2.P(3)^2.y(3)^2 + 2.n(1)^2.P(1).S(1).y(3)^2 + 2.n(1).n(2).P(2).S(1).y(3)^2 - n(1)^2.S(1)^2.y(3)^2
+ 2.n(1).n(2).P(1).S(2).y(3)^2 + 2.n(2)^2.P(2).S(2).y(3)^2 - 2.n(1).n(2).S(1).S(2).y(3)^2 - n(2)^2.S(2)^2.y(3)^2 + 2.n(1)^2.P(3).S(3).y(3)^2
+ 2.n(2)^2.P(3).S(3).y(3)^2 - n(1)^2.S(3)^2.y(3)^2 - n(2)^2.S(3)^2.y(3)^2)$^{(1/2)}$)/(n(2)^2.y(1)^2 + n(3)^2.y(1)^2 - 2.n(1).n(2).y(1).y(2) + n(1)^2.y(2)^2 + n(3)^2.y(2)^2
- 2.n(1).n(3).y(1).y(3) - 2.n(2).n(3).y(2).y(3) + n(1)^2.y(3)^2 + n(2)^2.y(3)^2)/2

Appendix C

Symbolic formulation of the Jacobian matrix for the rotary actuated three dof mechanism

C.1 The calculation of the pointing rates

The Jacobian matrix **J** relating the pointing rates $\dot{\theta}_2, \dot{\theta}_{3a}$ of the platform to the actuated (independent) joint rates $\dot{\beta} = (\dot{\beta}_1, \dot{\beta}_2, \dot{\beta}_3)$ is found by differentiating the closed form solution of the forward kinematics developed in Chapter 5. The forward Jacobian matrix relating the platform pointing rates to the independent joint rates for the three dof mechanism can be written as

$$\dot{\theta} = \mathbf{J}\dot{\beta}$$

where

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \theta_2}{\partial \beta_1} & \frac{\partial \theta_2}{\partial \beta_2} & \frac{\partial \theta_2}{\partial \beta_3} \\ \frac{\partial \theta_{3a}}{\partial \beta_1} & \frac{\partial \theta_{3a}}{\partial \beta_2} & \frac{\partial \theta_{3a}}{\partial \beta_3} \\ \frac{\partial |\boldsymbol{O}_p|}{\partial \beta_1} & \frac{\partial |\boldsymbol{O}_p|}{\partial \beta_2} & \frac{\partial |\boldsymbol{O}_p|}{\partial \beta_3} \end{bmatrix}$$

and the pointing rates are given by

$$\dot{\theta} = \begin{pmatrix} \dot{\theta}_2 \\ \dot{\theta}_{3a} \\ |\dot{\boldsymbol{O}}_p| \end{pmatrix}.$$

Note, for beam aiming applications the distance between the centre of the base and the centre of the platform $|O_p|$ is a redundant freedom.

C.1.1 Calculation of $\frac{\partial \theta_2}{\partial \beta_n}$

Recall from section 5.2.3 that the platform pointing angle θ_2 can be written as

$$\theta_2 = 2\cos^{-1}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right).$$

Partially differentiating θ_2 with respect to β_n (n = 1,2,3) gives

$$\frac{\partial \theta_2}{\partial \beta_n} = \frac{-2}{\sqrt{1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2}} \frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right) \tag{C.1}$$

and

$$\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right) = \frac{\frac{\partial O_{p(3)}}{\partial \beta_n}|\boldsymbol{O}_p| - O_{p(3)}\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_n}}{|\boldsymbol{O}_p|^2} \tag{C.2}$$

and

$$\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_n} = \frac{O_p^T \frac{\partial \boldsymbol{O}_p}{\partial \beta_n}}{|\boldsymbol{O}_p|}. \tag{C.3}$$

Recall from section 5.2.3 that the platform position vector $\boldsymbol{O}_p$ can be written as

$$\boldsymbol{O}_p = 2\,\boldsymbol{B}_1^T \hat{\boldsymbol{w}}_1 \hat{\boldsymbol{w}}_1.$$

Partially differentiating $\boldsymbol{O}_p$ with respect to β_n gives

$$\frac{\partial \boldsymbol{O}_p}{\partial \beta_n} = 2\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T \frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)\hat{\boldsymbol{w}}_1 + 2\left(\boldsymbol{B}_1^T \hat{\boldsymbol{w}}_1\right)\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n} \tag{C.4}$$

and partially differentiating the unit vector $\hat{\boldsymbol{w}}_1$ with respect to β_n gives

$$\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n} = \frac{\partial}{\partial \beta_n}\left(\frac{\boldsymbol{w}_1}{|\boldsymbol{w}_1|}\right) = \frac{\frac{\partial \boldsymbol{w}_1}{\partial \beta_n}|\boldsymbol{w}_1| - \boldsymbol{w}_1 \frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}}{|\boldsymbol{w}_1|^2} \tag{C.5}$$

where

$$\frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n} = \frac{\boldsymbol{w}_1^T \frac{\partial \boldsymbol{w}_1}{\partial \beta_n}}{|\boldsymbol{w}_1|}. \tag{C.6}$$

Recall from section 5.2.3 that the vector $\boldsymbol{w}_1$ can be expressed as

$$\boldsymbol{w}_1 = \overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3}.$$

Partially differentiating the vector $\boldsymbol{w}_1$ with respect to β_n gives

$$\frac{\partial \boldsymbol{w}_1}{\partial \beta_n} = \overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n} + \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_n} \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3}. \tag{C.7}$$

Partially differentiating the vectors $\overline{\boldsymbol{B}_1\boldsymbol{B}_2}$ and $\overline{\boldsymbol{B}_1\boldsymbol{B}_3}$ with respect to β_n gives

$$\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_m}}{\partial \beta_n} = \frac{\partial \boldsymbol{B}_m}{\partial \beta_n} - \frac{\partial \boldsymbol{B}_1}{\partial \beta_n}. \qquad \text{m=2,3} \tag{C.8}$$

From section 5.2.1 the vector B_k can be expressed as

$$B_k = {}^0\mathrm{T}_k\, {}^kB_k. \qquad \text{k= 1,2,3}$$

Partially differentiating the above with respect to β_n

$$\frac{\partial B_k}{\partial \beta_n} = {}^0\mathrm{T}_k \frac{\partial {}^kB_k}{\partial \beta_n} + \frac{\partial {}^0\mathrm{T}_k}{\partial \beta_n} {}^kB_k. \tag{C.9}$$

where

$$\frac{\partial {}^kB_k}{\partial \beta_n} = \begin{pmatrix} l_k \sin\beta_k \\ 0 \\ l_k \cos\beta_k \\ 1 \end{pmatrix} \quad \text{for n = k} \qquad \text{and} \qquad \frac{\partial {}^kB_k}{\partial \beta_n} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{for n} \neq \text{k} \tag{C.10}$$

and partially differentiating the transformation matrices ${}^0\mathrm{T}_k$ with respect to β_n gives

$$\frac{\partial {}^0\mathrm{T}_k}{\partial \beta_n} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{C.11}$$

C.1.2 Calculation of $\frac{\partial \theta_{3a}}{\partial \beta_n}$

Recall from section 5.2.3 that the platform pointing angle θ_{3a} can be written as

$$\theta_{3a} = \mathrm{atan2}\left(O_{p(2)}, O_{p(1)}\right).$$

Partially differentiating θ_{3a} with respect to β_n gives

$$\frac{\partial \theta_{3a}}{\partial \beta_n} = \frac{1}{1 + \left(\dfrac{O_{p(2)}}{O_{p(1)}}\right)^2} \frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right) \tag{C.12}$$

where

$$\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right) = \frac{\dfrac{\partial O_{p(2)}}{\partial \beta_n} O_{p(1)} - O_{p(2)} \dfrac{\partial O_{p(1)}}{\partial \beta_n}}{\left(O_{p(1)}\right)^2} \tag{C.13}$$

and $\frac{\partial O_{p(1)}}{\partial \beta_n}$ is given in equation (C.4).

C.2 The calculation of the time derivative of the pointing rates

The time derivative of the pointing rates can be re written as

$$\frac{d^2\theta_i}{dt^2} = \frac{d}{dt}\sum_{j=1}^{3}\frac{\partial\theta_i}{\partial\beta_j}\frac{d\beta_j}{dt} \qquad \text{i = 2, 3a.}$$

Using the product rule the above can be rewritten as

$$\frac{d^2\theta_i}{dt^2} = \sum_{j=1}^{3}\frac{d}{dt}\left(\frac{\partial\theta_i}{\partial\beta_j}\right)\frac{d\beta_j}{dt} + \frac{\partial\theta_i}{\partial\beta_j}\frac{d}{dt}\left(\frac{d\beta_j}{dt}\right)$$

$$= \sum_{k=1}^{3}\sum_{j=1}^{3}\left(\frac{\partial}{\partial\beta_k}\left(\frac{\partial\theta_i}{\partial\beta_j}\right)\frac{d\beta_k}{dt}\right)\frac{d\beta_j}{dt} + \sum_{j=1}^{3}\frac{\partial\theta_i}{\partial\beta_j}\frac{d}{dt}\left(\frac{d\beta_j}{dt}\right)$$

$$= \sum_{k=1}^{3}\sum_{j=1}^{3}\frac{\partial^2\theta_i}{\partial\beta_k\partial\beta_j}\frac{d\beta_k}{dt}\frac{d\beta_j}{dt} + \sum_{j=1}^{3}\frac{\partial\theta_i}{\partial\beta_j}\frac{d}{dt}\left(\frac{d\beta_j}{dt}\right).$$

In matrix notation the time derivative of the pointing rates can be written as

$$\frac{d^2\theta_2}{dt^2} = \begin{pmatrix}\frac{d\beta_1}{dt} & \frac{d\beta_2}{dt} & \frac{d\beta_3}{dt}\end{pmatrix}\underbrace{\begin{bmatrix}\frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_2}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_2}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_2}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_2}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_2}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_2}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_2}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_2}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_2}{\partial\beta_3}\right)\end{bmatrix}}_{\mathbf{H}_1}\begin{pmatrix}\frac{d\beta_1}{dt}\\ \frac{d\beta_2}{dt}\\ \frac{d\beta_3}{dt}\end{pmatrix} + \begin{pmatrix}\frac{\partial\theta_2}{\partial\beta_1} & \frac{\partial\theta_2}{\partial\beta_2} & \frac{\partial\theta_2}{\partial\beta_3}\end{pmatrix}\begin{pmatrix}\frac{d^2\beta_1}{dt^2}\\ \frac{d^2\beta_2}{dt^2}\\ \frac{d^2\beta_3}{dt^2}\end{pmatrix},$$

$$\frac{d^2\theta_{3a}}{dt^2} = \begin{pmatrix}\frac{d\beta_1}{dt} & \frac{d\beta_2}{dt} & \frac{d\beta_3}{dt}\end{pmatrix}\underbrace{\begin{bmatrix}\frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_{3a}}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_{3a}}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial\theta_{3a}}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_{3a}}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_{3a}}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial\theta_{3a}}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_{3a}}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_{3a}}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial\theta_{3a}}{\partial\beta_3}\right)\end{bmatrix}}_{\mathbf{H}_2}\begin{pmatrix}\frac{d\beta_1}{dt}\\ \frac{d\beta_2}{dt}\\ \frac{d\beta_3}{dt}\end{pmatrix} + \begin{pmatrix}\frac{\partial\theta_{3a}}{\partial\beta_1} & \frac{\partial\theta_{3a}}{\partial\beta_2} & \frac{\partial\theta_{3a}}{\partial\beta_3}\end{pmatrix}\begin{pmatrix}\frac{d^2\beta_1}{dt^2}\\ \frac{d^2\beta_2}{dt^2}\\ \frac{d^2\beta_3}{dt^2}\end{pmatrix}.$$

In a similar fashion, the time derivative of the rate $|\dot{\boldsymbol{O}}_p|$ can be written as

$$\frac{d^2|\boldsymbol{O}_p|}{dt^2} = \begin{pmatrix}\frac{d\beta_1}{dt} & \frac{d\beta_2}{dt} & \frac{d\beta_3}{dt}\end{pmatrix}\underbrace{\begin{bmatrix}\frac{\partial}{\partial\beta_1}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_1}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_2}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_2}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_3}\right)\\ \frac{\partial}{\partial\beta_3}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_1}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_2}\right) & \frac{\partial}{\partial\beta_3}\left(\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_3}\right)\end{bmatrix}}_{\mathbf{H}_3}\begin{pmatrix}\frac{d\beta_1}{dt}\\ \frac{d\beta_2}{dt}\\ \frac{d\beta_3}{dt}\end{pmatrix} + \begin{pmatrix}\frac{\partial|\boldsymbol{O}_p|}{\partial\beta_1} & \frac{\partial|\boldsymbol{O}_p|}{\partial\beta_2} & \frac{\partial|\boldsymbol{O}_p|}{\partial\beta_3}\end{pmatrix}\begin{pmatrix}\frac{d^2\beta_1}{dt^2}\\ \frac{d^2\beta_2}{dt^2}\\ \frac{d^2\beta_3}{dt^2}\end{pmatrix}.$$

The three equations above can be combined and written as

$$\ddot{\theta} = \dot{\mathbf{J}}\dot{\beta} + \mathbf{J}\ddot{\beta},$$

where the time derivative of the Jacobian matrix is given by

$$\dot{\mathbf{J}} = \begin{bmatrix} \dot{\beta}^T \mathbf{H}_1 \\ \dot{\beta}^T \mathbf{H}_2 \\ \dot{\beta}^T \mathbf{H}_3 \end{bmatrix}.$$

The terms needed to formulate the $\mathbf{H}_n$ matrices are derived in the following sections.

C.2.1 Calculation of $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \theta_2}{\partial \beta_n}\right)$

Partially differentiating $\frac{\partial \theta_2}{\partial \beta_n}$ with respect to β_m gives

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \theta_2}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{-2}{\sqrt{1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2}} \frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)\right)$$

$$= \left(\frac{-1}{\sqrt{1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2}}\right) \frac{\partial}{\partial \beta_m}\left(\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)\right) + \frac{\partial}{\partial \beta_m}\left(\frac{-1}{\sqrt{1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2}}\right) \frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right). \quad \text{(C.14)}$$

The term $\frac{\partial}{\partial \beta_m}\left(\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)\right)$ from equation (C.14) will now be solved. Differentiating equation (C.2) w.r.t. $\frac{\partial}{\partial \beta_m}$ gives

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(3)}}{\partial \beta_n}|\boldsymbol{O}_p| - O_{p(3)}\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_n}}{|\boldsymbol{O}_p|^2}\right)$$

$$= \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(3)}}{\partial \beta_n}|\boldsymbol{O}_p|}{|\boldsymbol{O}_p|^2}\right) - \frac{\partial}{\partial \beta_m}\left(\frac{O_{p(3)}\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_n}}{|\boldsymbol{O}_p|^2}\right). \quad \text{(C.15)}$$

The first term of equation (C.15) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(3)}}{\partial \beta_n}|\boldsymbol{O}_p|}{|\boldsymbol{O}_p|^2}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(3)}}{\partial \beta_n}}{|\boldsymbol{O}_p|}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(\frac{\partial O_{p(3)}}{\partial \beta_n}\right)|\boldsymbol{O}_p| - \frac{\partial|\boldsymbol{O}_p|}{\partial \beta_m}\frac{\partial O_{p(3)}}{\partial \beta_n}}{|\boldsymbol{O}_p|^2}$$

where $\frac{\partial|\boldsymbol{O}_p|}{\partial \beta_m}$ is given in equation (C.3) and

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial O_{p(3)}}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(2\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)\hat{\boldsymbol{w}}_1 + 2(\boldsymbol{B}_1^T\hat{\boldsymbol{w}}_1)\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)$$

$$= \frac{\partial}{\partial \beta_m}\left(2\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)\hat{\boldsymbol{w}}_1\right) + \frac{\partial}{\partial \beta_m}\left(2(\boldsymbol{B}_1^T\hat{\boldsymbol{w}}_1)\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right). \qquad \text{(C.16)}$$

The first term of equation (C.16) is given by

$$\frac{\partial}{\partial \beta_m}\left(2\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)\hat{\boldsymbol{w}}_1\right) = 2\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right) + 2\hat{\boldsymbol{w}}_1\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right)$$

and

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1 + \boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1\right) + \frac{\partial}{\partial \beta_m}\left(\boldsymbol{B}_1^T\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_n}\right) \qquad \text{(C.17)}$$

where the first term of equation (C.17) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\hat{\boldsymbol{w}}_1\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\right)\hat{\boldsymbol{w}}_1 + \left(\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}\right)\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_m}.$$

The terms $\frac{\partial \hat{\boldsymbol{w}}_1}{\partial \beta_m}$ and $\frac{\partial \boldsymbol{B}_1^T}{\partial \beta_n}$ are given in equations (C.5) and (C.9) respectively and

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left({}^0\mathbf{T}_k\frac{\partial {}^k\boldsymbol{B}_k}{\partial \beta_n} + \frac{\partial {}^0\mathbf{T}_k}{\partial \beta_n}{}^k\boldsymbol{B}_k\right) \qquad \text{k =1,2,3}$$

$$= \frac{\partial}{\partial \beta_m}\left({}^0\mathbf{T}_k\frac{\partial {}^k\boldsymbol{B}_k}{\partial \beta_n}\right) + \frac{\partial}{\partial \beta_m}\left(\frac{\partial {}^0\mathbf{T}_k}{\partial \beta_n}{}^k\boldsymbol{B}_k\right) \qquad \text{(C.18)}$$

where the first term of equation (C.18) is given by

$$\frac{\partial}{\partial \beta_m}\left({}^0\mathbf{T}_k\frac{\partial {}^k\boldsymbol{B}_k}{\partial \beta_n}\right) = \frac{\partial {}^0\mathbf{T}_k}{\partial \beta_m}\frac{\partial {}^k\boldsymbol{B}_k}{\partial \beta_n} + {}^0\mathbf{T}_k\frac{\partial}{\partial \beta_m}\left(\frac{\partial {}^k\boldsymbol{B}_k}{\partial \beta_n}\right).$$

The terms $\frac{\partial^k \boldsymbol{B}_k}{\partial \beta_n}$ and $\frac{\partial^0 \mathbf{T}_k}{\partial \beta_m}$ are given in equations (C.10) and (C.11) respectively and

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial^k \boldsymbol{B}_k}{\partial \beta_n}\right)=\begin{pmatrix} -l_k \cos\beta_k \\ 0 \\ l_k \sin\beta_k \\ 1 \end{pmatrix} \quad \text{for m = k} \qquad \text{and} \quad \frac{\partial}{\partial \beta_m}\left(\frac{\partial^k \boldsymbol{B}_k}{\partial \beta_n}\right)=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{for m} \neq \text{k}.$$

The second term of equation (C.18) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial^0 \mathbf{T}_k}{\partial \beta_n}\,{}^k\boldsymbol{B}_k\right)=\frac{\partial}{\partial \beta_m}\left(\frac{\partial^0 \mathbf{T}_k}{\partial \beta_n}\right){}^k\boldsymbol{B}_k + \frac{\partial^0 \mathbf{T}_k}{\partial \beta_m}\frac{\partial^k \boldsymbol{B}_k}{\partial \beta_n}$$

where

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial^0 \mathbf{T}_k}{\partial \beta_n}\right)=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The second term of equation (C.17) is given by

$$\frac{\partial}{\partial \beta_m}\left(\boldsymbol{B}_l^T \frac{\partial \hat{\boldsymbol{w}}_l}{\partial \beta_n}\right) = \frac{\partial \boldsymbol{B}_l^T}{\partial \beta_m}\frac{\partial \hat{\boldsymbol{w}}_l}{\partial \beta_n} + \boldsymbol{B}_l^T \frac{\partial}{\partial \beta_m}\left(\frac{\partial \hat{\boldsymbol{w}}_l}{\partial \beta_n}\right)$$

where

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \hat{\boldsymbol{w}}_l}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}|\boldsymbol{w}_l| - \boldsymbol{w}_l \frac{\partial |\boldsymbol{w}_l|}{\partial \beta_n}}{|\boldsymbol{w}_l|^2}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}}{|\boldsymbol{w}_l|}\right) - \frac{\partial}{\partial \beta_m}\left(\boldsymbol{w}_l \frac{\frac{\partial |\boldsymbol{w}_l|}{\partial \beta_n}}{|\boldsymbol{w}_l|^2}\right). \quad \text{(C.19)}$$

The first term of equation (C.19) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}}{|\boldsymbol{w}_l|}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}\right)|\boldsymbol{w}_l| - \frac{\partial \boldsymbol{w}_l}{\partial \beta_n}\frac{\partial |\boldsymbol{w}_l|}{\partial \beta_m}}{|\boldsymbol{w}_l|^2}.$$

The terms $\frac{\partial |\boldsymbol{w}_l|}{\partial \beta_m}$, $\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}$ are given in equations (C.6) and (C.7) respectively and

$$\begin{aligned}\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{w}_l}{\partial \beta_n}\right) &= \frac{\partial}{\partial \beta_m}\left(\overline{\boldsymbol{B}_l \boldsymbol{B}_2} \times \frac{\partial \overline{\boldsymbol{B}_l \boldsymbol{B}_3}}{\partial \beta_n} + \frac{\partial \overline{\boldsymbol{B}_l \boldsymbol{B}_2}}{\partial \beta_n} \times \overline{\boldsymbol{B}_l \boldsymbol{B}_3}\right) \\ &= \frac{\partial}{\partial \beta_m}\left(\overline{\boldsymbol{B}_l \boldsymbol{B}_2} \times \frac{\partial \overline{\boldsymbol{B}_l \boldsymbol{B}_3}}{\partial \beta_n}\right) + \frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_l \boldsymbol{B}_2}}{\partial \beta_n} \times \overline{\boldsymbol{B}_l \boldsymbol{B}_3}\right). \quad \text{(C.20)}\end{aligned}$$

The first term of equation (C.20) is given by

$$\frac{\partial}{\partial \beta_m}\left(\overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n}\right) = \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_m} \times \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n} + \overline{\boldsymbol{B}_1\boldsymbol{B}_2} \times \frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n}\right).$$

The terms $\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_m}$, $\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n}$ are given in equation (C.8) and

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_3}{\partial \beta_n} - \frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_3}{\partial \beta_n}\right) - \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right).$$

The terms $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_3}{\partial \beta_n}\right)$, $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right)$ are given in equation (C.18).

The second term of equation (C.20) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_n} \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_n}\right) \times \overline{\boldsymbol{B}_1\boldsymbol{B}_3} + \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_n} \times \frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_3}}{\partial \beta_m}$$

where

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \overline{\boldsymbol{B}_1\boldsymbol{B}_2}}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_2}{\partial \beta_n} - \frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_2}{\partial \beta_n}\right) - \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_1}{\partial \beta_n}\right).$$

The term $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{B}_2}{\partial \beta_n}\right)$ is given in equation (C.18).

The second term of equation (C.19) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{\boldsymbol{w}_1 \frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}}{|\boldsymbol{w}_1|^2}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(\boldsymbol{w}_1 \frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}\right)|\boldsymbol{w}_1|^2 - \boldsymbol{w}_1 \frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}\frac{\partial |\boldsymbol{w}_1|^2}{\partial \beta_m}}{|\boldsymbol{w}_1|^4} \tag{C.21}$$

where the term $\frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}$ is given in equation (C.6) and

$$\frac{\partial |\boldsymbol{w}_1|^2}{\partial \beta_m} = 2|\boldsymbol{w}_1|\frac{\partial |\boldsymbol{w}_1|}{\partial \beta_m}$$

and

$$\frac{\partial}{\partial \beta_m}\left(\boldsymbol{w}_1 \frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}\right) = \frac{\partial \boldsymbol{w}_1}{\partial \beta_m}\frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n} + \boldsymbol{w}_1 \frac{\partial}{\partial \beta_m}\left(\frac{\partial |\boldsymbol{w}_1|}{\partial \beta_n}\right),$$

where

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial |w_1|}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{w_1^T \frac{\partial w_1}{\partial \beta_n}}{|w_1|}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(w_1^T \frac{\partial w_1}{\partial \beta_n}\right)|w_1| - w_1^T \frac{\partial w_1}{\partial \beta_n}\frac{\partial |w_1|}{\partial \beta_m}}{|w_1|^2}$$

$$= \frac{\frac{\partial}{\partial \beta_m}\left(w_1^T \frac{\partial w_1}{\partial \beta_n}\right)}{|w_1|} - \frac{w_1^T \frac{\partial w_1}{\partial \beta_n}\frac{\partial |w_1|}{\partial \beta_m}}{|w_1|^2}$$

and

$$\frac{\partial}{\partial \beta_m}\left(w_1^T \frac{\partial w_1}{\partial \beta_n}\right) = \frac{\partial w_1^T}{\partial \beta_m}\frac{\partial w_1}{\partial \beta_n} + w_1^T \frac{\partial}{\partial \beta_m}\left(\frac{\partial w_1}{\partial \beta_n}\right).$$

The term $\frac{\partial}{\partial \beta_m}\left(\frac{\partial w_1}{\partial \beta_n}\right)$ is given in equation (C.20).

The second term in equation (C.16) is given by

$$\frac{\partial}{\partial \beta_m}\left(2(B_1^T \hat{w}_1)\frac{\partial \hat{w}_1}{\partial \beta_n}\right) = 2\frac{\partial (B_1^T \hat{w}_1)}{\partial \beta_m}\frac{\partial \hat{w}_1}{\partial \beta_n} + 2 B_1^T \hat{w}_1 \frac{\partial}{\partial \beta_m}\left(\frac{\partial \hat{w}_1}{\partial \beta_n}\right).$$

The term $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \hat{w}_1}{\partial \beta_n}\right)$ is given by equation (C.19) and

$$\frac{\partial (B_1^T \hat{w}_1)}{\partial \beta_m} = \frac{\partial B_1^T}{\partial \beta_m}\hat{w}_1 + B_1^T \frac{\partial \hat{w}_1}{\partial \beta_m}.$$

The second term in equation (C.15) is given by

$$\frac{\partial}{\partial \beta_m}\left(\frac{O_{p(3)}\frac{\partial |O_p|}{\partial \beta_n}}{|O_p|^2}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(O_{p(3)}\frac{\partial |O_p|}{\partial \beta_n}\right)|O_p|^2 - O_{p(3)}\frac{\partial |O_p|}{\partial \beta_n}\frac{\partial |O_p|^2}{\partial \beta_m}}{|O_p|^4}$$

where

$$\frac{\partial}{\partial \beta_m}\left(O_{p(3)}\frac{\partial |O_p|}{\partial \beta_n}\right) = \frac{\partial O_{p(3)}}{\partial \beta_m}\frac{\partial |O_p|}{\partial \beta_n} + O_{p(3)}\frac{\partial}{\partial \beta_m}\left(\frac{\partial |O_p|}{\partial \beta_n}\right)$$

and

$$\frac{\partial |O_p|^2}{\partial \beta_m} = 2|O_p|\frac{\partial |O_p|}{\partial \beta_m},$$

where

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\boldsymbol{O}_p^T \frac{\partial \boldsymbol{O}_p}{\partial \beta_n}}{|\boldsymbol{O}_p|}\right)$$

$$= \frac{\frac{\partial}{\partial \beta_m}\left(\boldsymbol{O}_p^T \frac{\partial \boldsymbol{O}_p}{\partial \beta_n}\right)|\boldsymbol{O}_p| - \boldsymbol{O}_p^T \frac{\partial \boldsymbol{O}_p}{\partial \beta_n}\frac{\partial |\boldsymbol{O}_p|}{\partial \beta_m}}{|\boldsymbol{O}_p|^2}$$

and

$$\frac{\partial}{\partial \beta_m}\left(\boldsymbol{O}_p^T \frac{\partial \boldsymbol{O}_p}{\partial \beta_n}\right) = \frac{\partial \boldsymbol{O}_p^T}{\partial \beta_m}\frac{\partial \boldsymbol{O}_p}{\partial \beta_n} + \boldsymbol{O}_p^T \frac{\partial}{\partial \beta_m}\left(\frac{\partial \boldsymbol{O}_p}{\partial \beta_n}\right).$$

The first part of the second term in equation (C.14) can be written as

$$\frac{\partial}{\partial \beta_m}\left(\frac{-1}{\sqrt{1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2}}\right) = -\frac{\partial}{\partial \beta_m}\left(1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2\right)^{-\frac{1}{2}} = \frac{1}{2}\left(1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2\right)^{-\frac{3}{2}} \frac{\partial}{\partial \beta_m}\left(1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2\right)$$

where

$$\frac{\partial}{\partial \beta_m}\frac{\partial}{\partial \beta_m}\left(1-\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2\right) = 0 - \frac{\partial}{\partial \beta_m}\left(\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)^2\right) = -2\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right)\frac{\partial}{\partial \beta_m}\left(\frac{O_{p(3)}}{|\boldsymbol{O}_p|}\right).$$

C.2.2 Calculation of $\frac{\partial}{\partial \beta_m}\left(\frac{\partial \theta_{3a}}{\partial \beta_n}\right)$

Partially differentiating $\frac{\partial \theta_{3a}}{\partial \beta_n}$ with respect to β_m gives

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial \theta_{3a}}{\partial \beta_n}\right) = \frac{\partial}{\partial \beta_m}\left(\frac{1}{1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2}\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right)\right)$$

$$= \frac{\partial}{\partial \beta_m}\left(\frac{1}{1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2}\right)\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right) + \frac{1}{1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2}\frac{\partial}{\partial \beta_m}\left(\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right)\right). \quad \text{(C.22)}$$

The term $\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right)$ is given in equation (C.13) and

$$\frac{\partial}{\partial \beta_m}\left(\frac{1}{1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2}\right) = \frac{\partial}{\partial \beta_m}\left(1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2\right)^{-1} = -\left(1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2\right)^{-2}\frac{\partial}{\partial \beta_m}\left(1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2\right)$$

where

$$\frac{\partial}{\partial \beta_m}\left(1+\left(\frac{O_{p(2)}}{O_{p(1)}}\right)^2\right) = 0 + 2\frac{O_{p(2)}}{O_{p(1)}}\frac{\partial}{\partial \beta_m}\left(\frac{O_{p(2)}}{O_{p(1)}}\right).$$

The second part of the second term of equation (C.22) can be written as

$$\frac{\partial}{\partial \beta_m}\left(\frac{\partial}{\partial \beta_n}\left(\frac{O_{p(2)}}{O_{p(1)}}\right)\right) = \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(2)}}{\partial \beta_n}O_{p(1)} - O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}}{(O_{p(1)})^2}\right)$$

$$= \frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(2)}}{\partial \beta_n}}{O_{p(1)}}\right) - \frac{\partial}{\partial \beta_m}\left(\frac{O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}}{(O_{p(1)})^2}\right). \qquad \text{(C.23)}$$

The first term of equation (C.23) can be written as

$$\frac{\partial}{\partial \beta_m}\left(\frac{\frac{\partial O_{p(2)}}{\partial \beta_n}}{O_{p(1)}}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(\frac{\partial O_{p(2)}}{\partial \beta_n}\right)O_{p(1)} - \frac{\partial O_{p(2)}}{\partial \beta_n}\frac{\partial O_{p(1)}}{\partial \beta_m}}{(O_{p(1)})^2}.$$

The terms $\frac{\partial O_{p(2)}}{\partial \beta_n}$, $\frac{\partial O_{p(1)}}{\partial \beta_m}$, $\frac{\partial}{\partial \beta_m}\left(\frac{\partial O_{p(2)}}{\partial \beta_n}\right)$ are given in equations (C.4) and (C.16).

The second term of equation (C.23) can be written as

$$\frac{\partial}{\partial \beta_m}\left(\frac{O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}}{(O_{p(1)})^2}\right) = \frac{\frac{\partial}{\partial \beta_m}\left(O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}\right)(O_{p(1)})^2 - O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}\frac{\partial (O_{p(1)})^2}{\partial \beta_m}}{(O_{p(1)})^4} \qquad \text{(C.24)}$$

where

$$\frac{\partial}{\partial \beta_m}\left(O_{p(2)}\frac{\partial O_{p(1)}}{\partial \beta_n}\right) = \frac{\partial O_{p(2)}}{\partial \beta_m}\frac{\partial O_{p(1)}}{\partial \beta_n} + O_{p(2)}\frac{\partial}{\partial \beta_m}\left(\frac{\partial O_{p(1)}}{\partial \beta_n}\right)$$

and

$$\frac{\partial\left(O_{p(1)}\right)^2}{\partial \beta_m} = 2\,O_{p(1)}\frac{\partial O_{p(1)}}{\partial \beta_m}.$$

Appendix D

The kinematics of the platform

D.1. The calculation of the angular velocity of the platform

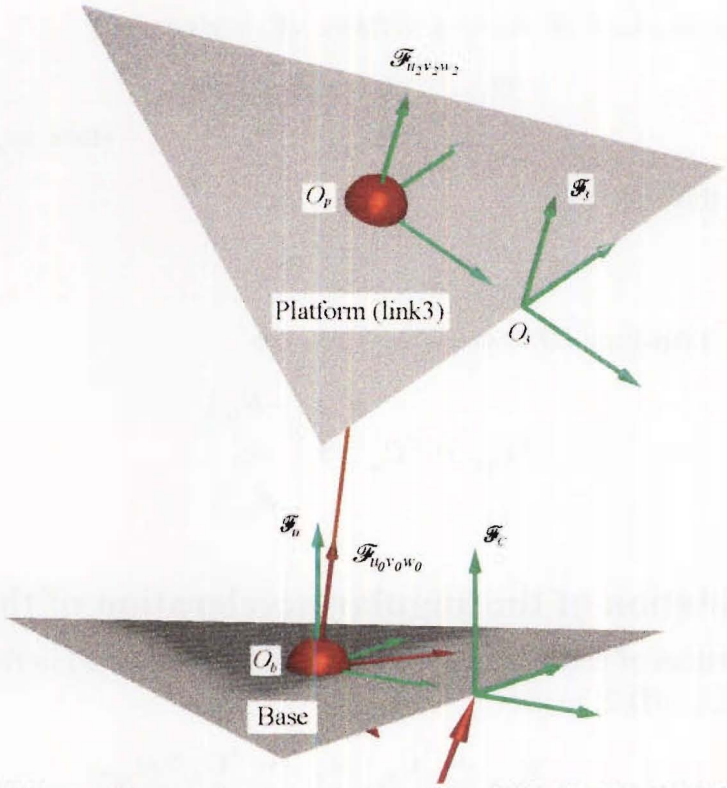


Figure D.1 The position and orientation of the platform frames $\mathcal{F}_{u_2v_2w_2}$ and $\mathcal{F}_3$.

In this appendix the angular velocity and acceleration of the platform (link 3) relative to the base frame $\mathcal{F}_C$ are calculated given the satellite motion. From section 5.2.3 the rotation matrices relating the coordinate frames $\mathcal{F}_0$ and $\mathcal{F}_{u_2v_2w_2}$ are given by

$$^{u_2}\mathbf{C}_0 = \mathbf{C}_3(-\theta_{3a})\,\mathbf{C}_2(\theta_2)\,\mathbf{C}_3(\theta_{3a}).$$

Using the relationship between the frames $\mathcal{F}_0$ and $\mathcal{F}_{u_2v_2w_2}$ above, the angular velocity of frame $\mathcal{F}_{u_2v_2w_2}$ relative to frame $\mathcal{F}_0$ expressed in frame $\mathcal{F}_{u_2v_2w_2}$ can be written as

$$^{u_2}\boldsymbol{\omega}_{u_2,0} = {}^{u_2}\mathbf{S}_0 \begin{pmatrix} -\dot{\theta}_{3a} \\ \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix} \qquad \text{(D.1)}$$

where

$$ {}^{u_2}\mathbf{S}_0 = \begin{bmatrix} 0 & s(-\theta_{3a}) & -c(-\theta_{3a})s\theta_2 \\ 0 & c(-\theta_{3a}) & s\theta_2 s(-\theta_{3a}) \\ 1 & 0 & c\theta_2 \end{bmatrix}. $$

The angular velocity of frame $\mathscr{F}_{u_2v_2w_2}$ relative to frame $\mathscr{F}_C$ is given by

$$ \begin{aligned} \underline{\boldsymbol{\omega}}_{u_2,C} &= \underline{\boldsymbol{\omega}}_{0,C} + \underline{\boldsymbol{\omega}}_{u_2,0} \\ &= \underline{\boldsymbol{\omega}}_{u_2,0}; \qquad \text{since } \underline{\boldsymbol{\omega}}_{0,C} = \mathbf{0} \end{aligned} $$

in component form this becomes

$$ {}^{u_2}\boldsymbol{\omega}_{u_2,C} = {}^{u_2}\boldsymbol{\omega}_{u_2,0}. $$

The angular velocity of frame $\mathscr{F}_3$ relative to frame $\mathscr{F}_C$ is given by

$$ \begin{aligned} \underline{\boldsymbol{\omega}}_{3,C} &= \underline{\boldsymbol{\omega}}_{0,C} + \underline{\boldsymbol{\omega}}_{u_2,0} + \underline{\boldsymbol{\omega}}_{3,u_2} \\ &= \underline{\boldsymbol{\omega}}_{u_2,0}; \qquad \text{since } \underline{\boldsymbol{\omega}}_{3,u_2} = \mathbf{0} \end{aligned} $$

in component form this becomes

$$ {}^{3}\boldsymbol{\omega}_{3,C} = {}^{3}\mathbf{C}_{u_2}\,{}^{u_2}\boldsymbol{\omega}_{u_2,0}. \tag{D.2} $$

Using equation (D.1) equation (D.2) can be rewritten as

$$ {}^{3}\boldsymbol{\omega}_{3,C} = {}^{3}\mathbf{C}_{u_2}\,{}^{u_2}\mathbf{S}_0 \begin{pmatrix} -\dot{\theta}_{3a} \\ \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix}. $$

D.2 The calculation of the angular acceleration of the platform

The angular acceleration of frame $\mathscr{F}_3$ relative to frame $\mathscr{F}_C$ expressed in frame $\mathscr{F}_3$ is found by differentiating equation (D.2) to get

$$ \begin{aligned} {}^{3}\dot{\boldsymbol{\omega}}_{3,C} &= {}^{3}\dot{\mathbf{C}}_{u_2}\,{}^{u_2}\boldsymbol{\omega}_{u_2,0} + {}^{3}\mathbf{C}_{u_2}\,{}^{u_2}\dot{\boldsymbol{\omega}}_{u_2,0} \\ &= {}^{3}\mathbf{C}_{u_2}\,{}^{u_2}\dot{\boldsymbol{\omega}}_{u_2,0} \qquad \text{since } {}^{3}\dot{\mathbf{C}}_{u_2} = \mathbf{O}. \end{aligned} $$

The angular acceleration of frame $\mathscr{F}_{u_2v_2w_2}$ relative to frame $\mathscr{F}_0$ expressed in frame $\mathscr{F}_{u_2v_2w_2}$ is found by differentiating equation (D.1) to get

$$ {}^{u_2}\dot{\boldsymbol{\omega}}_{u_2,0} = {}^{u_2}\dot{\mathbf{S}}_0 \begin{pmatrix} -\dot{\theta}_{3a} \\ \dot{\theta}_2 \\ \dot{\theta}_{3a} \end{pmatrix} + {}^{u_2}\mathbf{S}_0 \begin{pmatrix} -\ddot{\theta}_{3a} \\ \ddot{\theta}_2 \\ \ddot{\theta}_{3a} \end{pmatrix} $$

where

$$ {}^{u_2}\mathbf{S}_0 = \begin{bmatrix} 0 & -\dot{\theta}_{3a}c(-\theta_{3a}) & -\dot{\theta}_{3a}s(-\theta_{3a})s\theta_2 - \dot{\theta}_2 c(-\theta_{3a})c\theta_2 \\ 0 & \dot{\theta}_{3a}s(-\theta_{3a}) & \dot{\theta}_2 c\theta_2 s(-\theta_{3a}) - \dot{\theta}_{3a}s\theta_2 c(-\theta_{3a}) \\ 0 & 0 & -\dot{\theta}_2 s\theta_2 \end{bmatrix}. $$

Appendix E

The global formulation for the two dof prismatically actuated mechanism

E.1 Introduction

In this appendix the matrices used to formulate the dynamic equations for the two dof prismatically actuated mechanism are derived. A model of the mechanism is shown in figure E.1 with more detailed models of the coordinate systems shown in the figures following.

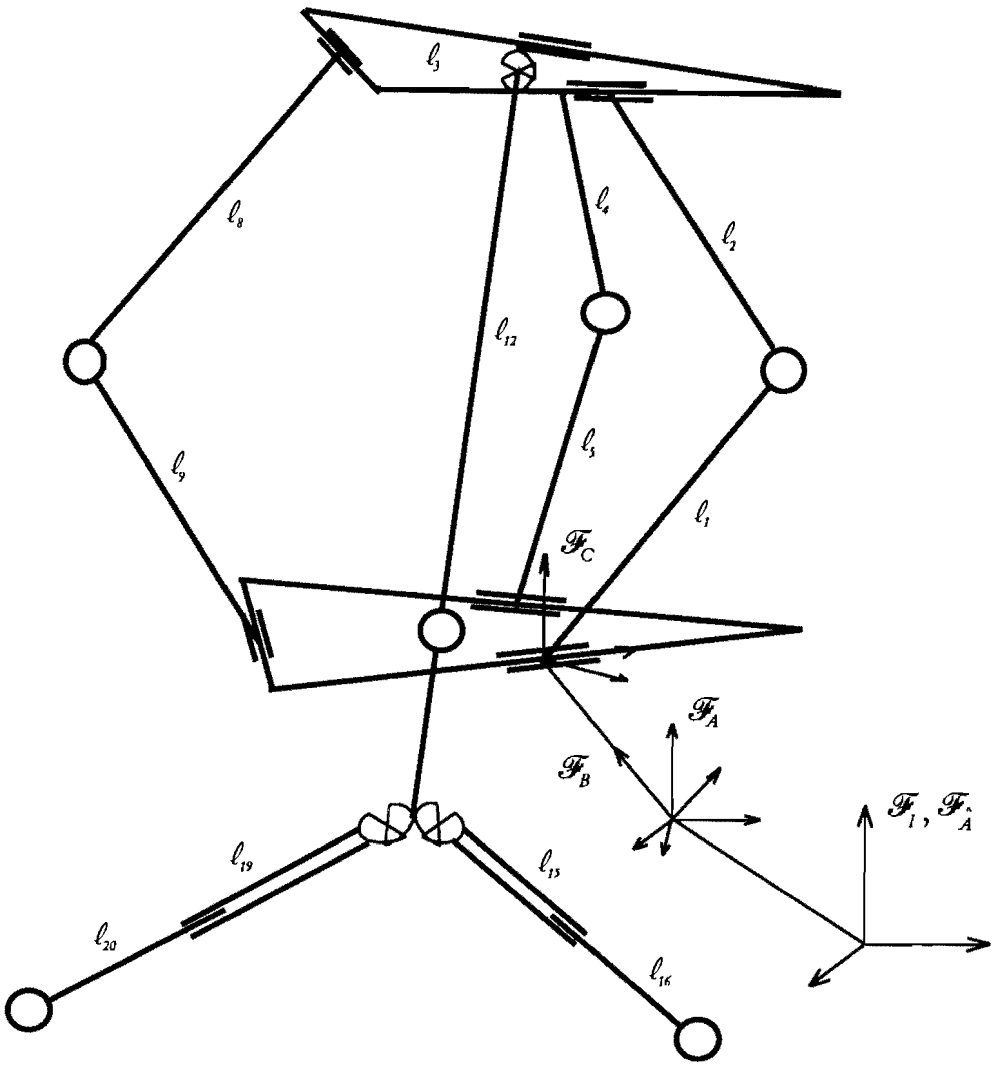


Figure E.1 Kinematic model of the two dof prismatically actuated mechanism.

E.2 Global interbody transformation matrix $\mathcal{T}$

The global interbody transformation matrix $\mathcal{T}$ for the mechanism (with the ship included) can be written as

$$\mathcal{T} = \left[\begin{array}{cccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{B}\mathcal{T}_{A} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{B}\mathcal{T}_{A} & {}^{C}\mathcal{T}_{B} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{1}\mathcal{T}_{A} & {}^{1}\mathcal{T}_{B} & {}^{1}\mathcal{T}_{C} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{2}\mathcal{T}_{A} & {}^{2}\mathcal{T}_{B} & {}^{2}\mathcal{T}_{C} & {}^{2}\mathcal{T}_{1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{3}\mathcal{T}_{A} & {}^{3}\mathcal{T}_{B} & {}^{3}\mathcal{T}_{C} & {}^{3}\mathcal{T}_{1} & {}^{3}\mathcal{T}_{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{4}\mathcal{T}_{A} & {}^{4}\mathcal{T}_{B} & {}^{4}\mathcal{T}_{C} & {}^{4}\mathcal{T}_{1} & {}^{4}\mathcal{T}_{2} & {}^{4}\mathcal{T}_{3} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{5}\mathcal{T}_{A} & {}^{5}\mathcal{T}_{B} & {}^{5}\mathcal{T}_{C} & {}^{5}\mathcal{T}_{1} & {}^{5}\mathcal{T}_{2} & {}^{5}\mathcal{T}_{3} & {}^{5}\mathcal{T}_{4} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{6}\mathcal{T}_{A} & {}^{6}\mathcal{T}_{B} & {}^{6}\mathcal{T}_{C} & {}^{6}\mathcal{T}_{1} & {}^{6}\mathcal{T}_{2} & {}^{6}\mathcal{T}_{3} & {}^{6}\mathcal{T}_{4} & {}^{6}\mathcal{T}_{5} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{8}\mathcal{T}_{A} & {}^{8}\mathcal{T}_{B} & {}^{8}\mathcal{T}_{C} & {}^{8}\mathcal{T}_{1} & {}^{8}\mathcal{T}_{2} & {}^{8}\mathcal{T}_{3} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{9}\mathcal{T}_{A} & {}^{9}\mathcal{T}_{B} & {}^{9}\mathcal{T}_{C} & {}^{9}\mathcal{T}_{1} & {}^{9}\mathcal{T}_{2} & {}^{9}\mathcal{T}_{3} & 0 & 0 & 0 & {}^{9}\mathcal{T}_{8} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{10}\mathcal{T}_{A} & {}^{10}\mathcal{T}_{B} & {}^{10}\mathcal{T}_{C} & {}^{10}\mathcal{T}_{1} & {}^{10}\mathcal{T}_{2} & {}^{10}\mathcal{T}_{3} & 0 & 0 & 0 & {}^{10}\mathcal{T}_{8} & {}^{10}\mathcal{T}_{9} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{12}\mathcal{T}_{A} & {}^{12}\mathcal{T}_{B} & {}^{12}\mathcal{T}_{C} & {}^{12}\mathcal{T}_{1} & {}^{12}\mathcal{T}_{2} & {}^{12}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{13}\mathcal{T}_{A} & {}^{13}\mathcal{T}_{B} & {}^{13}\mathcal{T}_{C} & {}^{13}\mathcal{T}_{1} & {}^{13}\mathcal{T}_{2} & {}^{13}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{13}\mathcal{T}_{12} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
{}^{15}\mathcal{T}_{A} & {}^{15}\mathcal{T}_{B} & {}^{15}\mathcal{T}_{C} & {}^{15}\mathcal{T}_{1} & {}^{15}\mathcal{T}_{2} & {}^{15}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{15}\mathcal{T}_{12} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
{}^{16}\mathcal{T}_{A} & {}^{16}\mathcal{T}_{B} & {}^{16}\mathcal{T}_{C} & {}^{16}\mathcal{T}_{1} & {}^{16}\mathcal{T}_{2} & {}^{16}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{16}\mathcal{T}_{12} & 0 & {}^{16}\mathcal{T}_{15} & 1 & 0 & 0 & 0 & 0 \\
{}^{17}\mathcal{T}_{A} & {}^{17}\mathcal{T}_{B} & {}^{17}\mathcal{T}_{C} & {}^{17}\mathcal{T}_{1} & {}^{17}\mathcal{T}_{2} & {}^{17}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{17}\mathcal{T}_{12} & 0 & {}^{17}\mathcal{T}_{15} & {}^{17}\mathcal{T}_{16} & 1 & 0 & 0 & 0 \\
{}^{19}\mathcal{T}_{A} & {}^{19}\mathcal{T}_{B} & {}^{19}\mathcal{T}_{C} & {}^{19}\mathcal{T}_{1} & {}^{19}\mathcal{T}_{2} & {}^{19}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{19}\mathcal{T}_{12} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
{}^{20}\mathcal{T}_{A} & {}^{20}\mathcal{T}_{B} & {}^{20}\mathcal{T}_{C} & {}^{20}\mathcal{T}_{1} & {}^{20}\mathcal{T}_{2} & {}^{20}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{20}\mathcal{T}_{12} & 0 & 0 & 0 & 0 & {}^{20}\mathcal{T}_{19} & 1 & 0 \\
{}^{21}\mathcal{T}_{A} & {}^{21}\mathcal{T}_{B} & {}^{21}\mathcal{T}_{C} & {}^{21}\mathcal{T}_{1} & {}^{21}\mathcal{T}_{2} & {}^{21}\mathcal{T}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & {}^{21}\mathcal{T}_{12} & 0 & 0 & 0 & 0 & {}^{21}\mathcal{T}_{19} & {}^{21}\mathcal{T}_{20} & 1
\end{array}\right]$$

The global interbody transformation matrix $\mathcal{T}_{mc}$ for the mechanism only is the partition that lies within the dashed lines. The interbody transformation matrices ${}^{n+1}\mathcal{T}_{n}$ are given by

$${}^{n+1}\mathcal{T}_{n} = \begin{bmatrix} {}^{n+1}\mathbf{C}_{n} & -{}^{n+1}\mathbf{C}_{n}\,{}^{n}\boldsymbol{\rho}^{\times}_{n,n+1} \\ \mathbf{0} & {}^{n+1}\mathbf{C}_{n} \end{bmatrix}.$$

E.2.1 Rotation matrices ${}^{n+1}\mathbf{C}_{n}$

In the following section the rotation matrices ${}^{n+1}\mathbf{C}_{n}$ are formulated for the mechanism. Referring to figure E.2 below

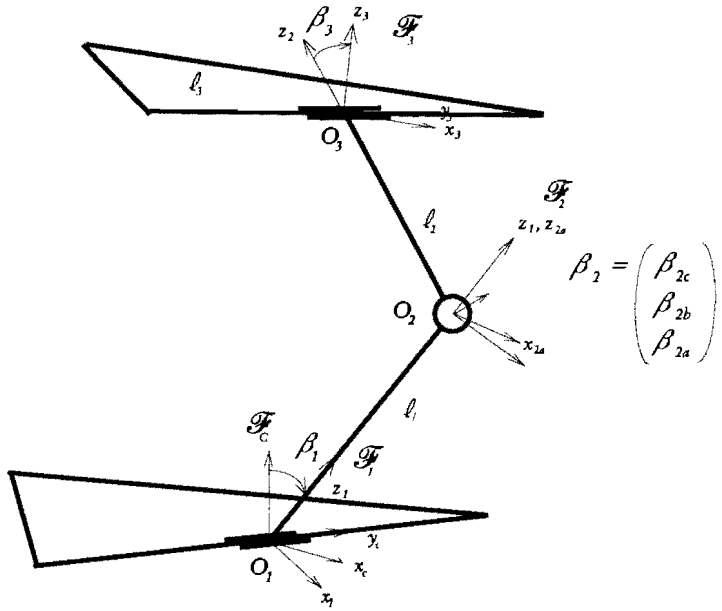


Figure E.2 Coordinate systems for the two dof prismatically actuated mechanism

the rotation matrices used to model the three joints positioned at O_1, O_2 and O_3 can be written as

$$
{}^{1}\mathbf{C}_{C} = \begin{bmatrix} c\beta_1 & 0 & -s\beta_1 \\ 0 & 1 & 0 \\ s\beta_1 & 0 & c\beta_1 \end{bmatrix},
$$

$$
{}^{2}\mathbf{C}_{1} = \begin{bmatrix} c\beta_{2c} & s\beta_{2c} & 0 \\ -s\beta_{2c} & c\beta_{2c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{2b} & 0 & -s\beta_{2b} \\ 0 & 1 & 0 \\ s\beta_{2b} & 0 & c\beta_{2b} \end{bmatrix} \begin{bmatrix} c\beta_{2a} & s\beta_{2a} & 0 \\ -s\beta_{2a} & c\beta_{2a} & 0 \\ 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{3}\mathbf{C}_{2} = \begin{bmatrix} c\beta_3 & 0 & -s\beta_3 \\ 0 & 1 & 0 \\ s\beta_3 & 0 & c\beta_3 \end{bmatrix}.
$$

Referring to figure E.3 below

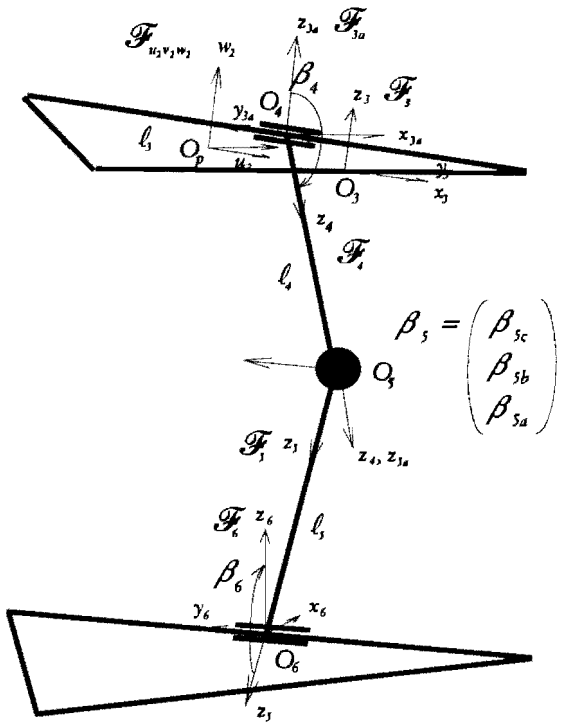


Figure E.3 Coordinate systems for the two dof prismatically actuated mechanism.

the rotation matrices used to model the transformation from joint O_3 to joint O_4 can be written as

$$
{}^{4}\mathbf{C}_{3} = {}^{4}\mathbf{C}_{3a}\,{}^{3a}\mathbf{C}_{u_2}\,{}^{u_2}\mathbf{C}_{3}
$$

where the matrices ${}^{u_2}\mathbf{C}_3$ and ${}^{3a}\mathbf{C}_{u_2}$ are formed from 3-2-1 sets (c.f. section 5.2.1) and

$$
{}^{4}\mathbf{C}_{3a} = \begin{bmatrix} c\beta_4 & 0 & -s\beta_4 \\ 0 & 1 & 0 \\ s\beta_4 & 0 & c\beta_4 \end{bmatrix}.
$$

The rotation matrices used to model the two joints positioned at O_5 and O_6 can be written as

$$
{}^{5}\mathbf{C}_{4} = \begin{bmatrix} c\beta_{5c} & s\beta_{5c} & 0 \\ -s\beta_{5c} & c\beta_{5c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{5b} & 0 & -s\beta_{5b} \\ 0 & 1 & 0 \\ s\beta_{5b} & 0 & c\beta_{5b} \end{bmatrix} \begin{bmatrix} c\beta_{5a} & s\beta_{5a} & 0 \\ -s\beta_{5a} & c\beta_{5a} & 0 \\ 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{6}\mathbf{C}_{5} = \begin{bmatrix} c\beta_{6} & 0 & -s\beta_{6} \\ 0 & 1 & 0 \\ s\beta_{6} & 0 & c\beta_{6} \end{bmatrix}.
$$

Referring to figure E.4 below

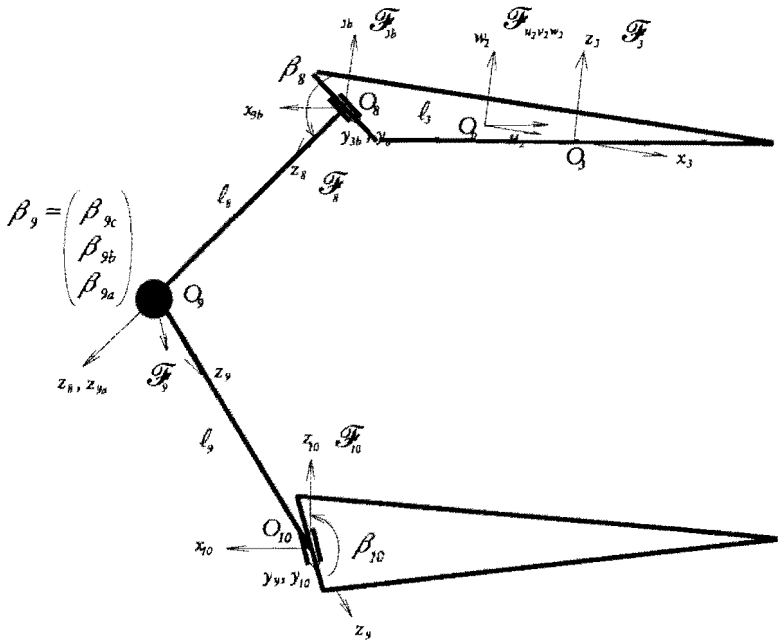


Figure E.4 Coordinate systems for the two dof prismatically actuated mechanism.

the rotation matrices used to model the transformation from joint O_3 to joint O_8 can be written as

$$
{}^{8}\mathbf{C}_{3} = {}^{8}\mathbf{C}_{3b}\,{}^{3b}\mathbf{C}_{u_2}\,{}^{u_2}\mathbf{C}_{3}
$$

where the matrices ${}^{u_2}\mathbf{C}_{3}$ and ${}^{3b}\mathbf{C}_{u_2}$ are formed from 3-2-1 sets (c.f. section 5.2.1) and

$$
{}^{8}\mathbf{C}_{3b} = \begin{bmatrix} c\beta_{8} & 0 & -s\beta_{8} \\ 0 & 1 & 0 \\ s\beta_{8} & 0 & c\beta_{8} \end{bmatrix}.
$$

The rotation matrices used to model the two joints positioned at O_9 and O_{10} can be written as

$$
{}^{9}\mathbf{C}_{8} = \begin{bmatrix} c\beta_{9c} & s\beta_{9c} & 0 \\ -s\beta_{9c} & c\beta_{9c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{9b} & 0 & -s\beta_{9b} \\ 0 & 1 & 0 \\ s\beta_{9b} & 0 & c\beta_{9b} \end{bmatrix} \begin{bmatrix} c\beta_{9a} & s\beta_{9a} & 0 \\ -s\beta_{9a} & c\beta_{9a} & 0 \\ 0 & 0 & 1 \end{bmatrix},
$$

$$ {}^{10}\mathbf{C}_9 = \begin{bmatrix} c\beta_{10} & 0 & -s\beta_{10} \\ 0 & 1 & 0 \\ s\beta_{10} & 0 & c\beta_{10} \end{bmatrix}. $$

Referring to figure E.5 below

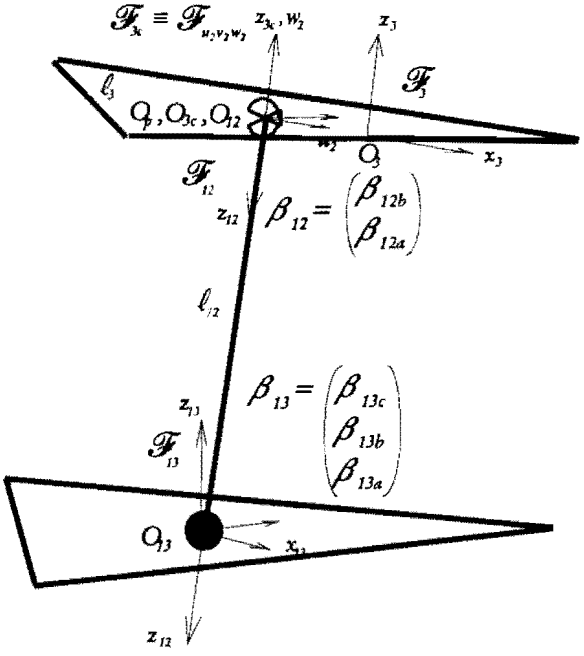


Figure E.5 Coordinate systems for the two dof prismatically actuated mechanism

the rotation matrices used to model the transformation from joint O_3 to joint O_{12} can be written as

$$ {}^{12}\mathbf{C}_3 = {}^{12}\mathbf{C}_{3c}\,{}^{3c}\mathbf{C}_3 $$

where the matrix ${}^{3c}\mathbf{C}_3$ is formed from a 3-2-1 set (c.f. section 5.2.1, note $\mathfrak{F}_{u_2v_2w_2} \equiv \mathfrak{F}_{3c}$) and

$$ {}^{12}\mathbf{C}_{3c} = \begin{bmatrix} c\beta_{12b} & 0 & -s\beta_{12b} \\ 0 & 1 & 0 \\ s\beta_{12b} & 0 & c\beta_{12b} \end{bmatrix} \begin{bmatrix} c\beta_{12a} & s\beta_{12a} & 0 \\ -s\beta_{12a} & c\beta_{12a} & 0 \\ 0 & 0 & 1 \end{bmatrix}. $$

The rotation matrices used to model the joint positioned at O_{13} can be written as

$$ {}^{13}\mathbf{C}_{12} = \begin{bmatrix} c\beta_{13c} & s\beta_{13c} & 0 \\ -s\beta_{13c} & c\beta_{13c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{13b} & 0 & -s\beta_{13b} \\ 0 & 1 & 0 \\ s\beta_{13b} & 0 & c\beta_{13b} \end{bmatrix} \begin{bmatrix} c\beta_{13a} & s\beta_{13a} & 0 \\ -s\beta_{13a} & c\beta_{13a} & 0 \\ 0 & 0 & 1 \end{bmatrix}. $$

Referring to figure E.6 below

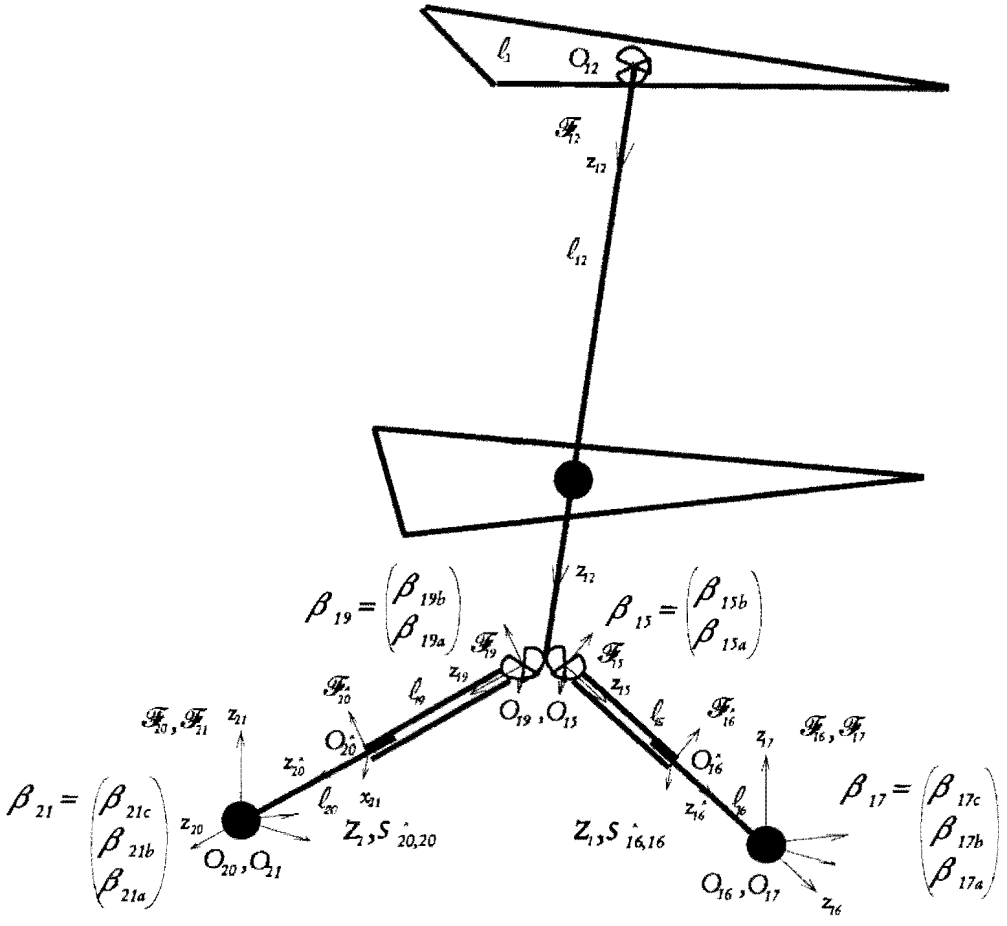


Figure E.6 Coordinate systems for the two dof prismatically actuated mechanism

the rotation matrices used to model the joint positioned at O_{15} can be written as

$$
{}^{15}\mathbf{C}_{12} = \begin{bmatrix} c\beta_{15b} & 0 & -s\beta_{15b} \\ 0 & 1 & 0 \\ s\beta_{15b} & 0 & c\beta_{15b} \end{bmatrix} \begin{bmatrix} c\beta_{15a} & s\beta_{15a} & 0 \\ -s\beta_{15a} & c\beta_{15a} & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

The rotation matrix used to model the joint positioned at O_{16} can be written as

$$
{}^{16}\mathbf{C}_{15} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

The rotation matrices used to model the joint positioned at O_{17} can be written as

$$
{}^{17}\mathbf{C}_{16} = \begin{bmatrix} c\beta_{17c} & s\beta_{17c} & 0 \\ -s\beta_{17c} & c\beta_{17c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{17b} & 0 & -s\beta_{17b} \\ 0 & 1 & 0 \\ s\beta_{17b} & 0 & c\beta_{17b} \end{bmatrix} \begin{bmatrix} c\beta_{17a} & s\beta_{17a} & 0 \\ -s\beta_{17a} & c\beta_{17a} & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

The rotation matrices used to model the joint positioned at O_{19} can be written as

$$^{19}\mathbf{C}_{12} = \begin{bmatrix} c\beta_{19b} & 0 & -s\beta_{19b} \\ 0 & 1 & 0 \\ s\beta_{19b} & 0 & c\beta_{19b} \end{bmatrix} \begin{bmatrix} c\beta_{19a} & s\beta_{19a} & 0 \\ -s\beta_{19a} & c\beta_{19a} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The rotation matrix used to model the joint positioned at $O_{2\hat{0}}$ can be written as

$$^{20}\mathbf{C}_{19} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The rotation matrices used to model the joint positioned at O_{21} can be written as

$$^{21}\mathbf{C}_{20} = \begin{bmatrix} c\beta_{21c} & s\beta_{21c} & 0 \\ -s\beta_{21c} & c\beta_{21c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta_{21b} & 0 & -s\beta_{21b} \\ 0 & 1 & 0 \\ s\beta_{21b} & 0 & c\beta_{21b} \end{bmatrix} \begin{bmatrix} c\beta_{21a} & s\beta_{21a} & 0 \\ -s\beta_{21a} & c\beta_{21a} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

E.2.2 Joint position vectors $^{n}\rho_{n,n+1}$

In the following section the joint position vectors are formulated for the mechanism.

$$^{A}\rho_{A,B} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$^{B}\rho_{B,C} = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix},$$

$$^{C}\rho_{C,1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$^{1}\rho_{1,2} = \begin{pmatrix} 0 \\ 0 \\ l_1 \end{pmatrix},$$

$$^{2}\rho_{2,3} = \begin{pmatrix} 0 \\ 0 \\ l_2 \end{pmatrix},$$

$$^{3}\rho_{3,4} = {}^{3}\mathbf{O}_4 - {}^{3}\mathbf{O}_3$$

$$^{4}\rho_{4,5} = \begin{pmatrix} 0 \\ 0 \\ l_4 \end{pmatrix},$$

$$^{5}\rho_{5,6} = \begin{pmatrix} 0 \\ 0 \\ l_5 \end{pmatrix},$$

$$^{3}\rho_{3,8} = {}^{3}\mathbf{O}_8 - {}^{3}\mathbf{O}_3,$$

$$^{8}\rho_{8,9} = \begin{pmatrix} 0 \\ 0 \\ l_8 \end{pmatrix},$$

$$^{9}\rho_{9,10} = \begin{pmatrix} 0 \\ 0 \\ l_9 \end{pmatrix},$$

$$^{3}\rho_{3,12} = {}^{3}\mathbf{O}_{12} - {}^{3}\mathbf{O}_3,$$

$$^{12}\rho_{12,15} = \begin{pmatrix} 0 \\ 0 \\ l_{12} + l_{12hookes} \end{pmatrix},$$

$$^{15}\rho_{15,16} = \begin{pmatrix} 0 \\ 0 \\ l_{15} \end{pmatrix} + {}^{16}\mathbf{C}_{15}^{T} \begin{pmatrix} 0 \\ 0 \\ s_{1\hat{6}.16} \end{pmatrix},$$

$$^{16}\rho_{16,17} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$^{12}\rho_{12,19} = \begin{pmatrix} 0 \\ 0 \\ l_{12} + l_{12hookes} \end{pmatrix},$$

$$^{19}\rho_{19,20} = \begin{pmatrix} 0 \\ 0 \\ l_{19} \end{pmatrix} + {}^{20}\mathbf{C}_{19}^{T} \begin{pmatrix} 0 \\ 0 \\ s_{2\hat{0},20} \end{pmatrix},$$

$$^{20}\rho_{20,21} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

E.3 Global transformation matrix $\mathcal{G}$

The global transformation matrix $\mathcal{G}$ for the mechanism (with the ship included) can be written as

$$\mathcal{G} = \begin{bmatrix} {}^{A}\mathcal{G}_{A,A} & & & & & & & & & & & & & & & & & & & & 0 \\ & {}^{B}\mathcal{G}_{B,B} & & & & & & & & & & & & & & & & & & & \\ & & {}^{C}\mathcal{G}_{C,C} & & & & & & & & & & & & & & & & & & \\ & & & {}^{1}\mathcal{G}_{1,1} & & & & & & & & & & & & & & & & & \\ & & & & {}^{2}\mathcal{G}_{2,2} & & & & & & & & & & & & & & & & \\ & & & & & {}^{3}\mathcal{G}_{3,3} & & & & & & & & & & & & & & & \\ & & & & & & {}^{4}\mathcal{G}_{4,4} & & & & & & & & & & & & & & \\ & & & & & & & \ddots & & & & & & & & & & & & \\ & & & & & & & & {}^{20}\mathcal{G}_{20,20} & & & & & & & & & & \\ 0 & & & & & & & & & {}^{21}\mathcal{G}_{21,21} & & & & & & & & & & \end{bmatrix}$$

The global transformation matrix $\mathcal{G}_{mc}$ for the mechanism only is the partition that lies within the dashed lines. The transformation matrices ${}^{n+1}\mathcal{G}_{n+1,n+1}$ are given by

$${}^{n+1}\mathcal{G}_{n+1,n+1} = \begin{bmatrix} \mathbf{1} & -{}^{n+1}\boldsymbol{s}^{\times}_{\hat{n+1},n+1} \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$$

where the displacement vectors are given by

$${}^{n+1}\boldsymbol{s}_{\hat{n+1},n+1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \text{for n = B,C,1,2,3,4,5,6,8,9,10,12,13,15,17,19,21}$$

and

$${}^{A}\boldsymbol{s}_{\hat{A}A} = \begin{pmatrix} A_{t_1}\, sin(\varepsilon_{t_1}\, t + \phi_{t_1}) \\ A_{t_2}\, sin(\varepsilon_{t_2}\, t + \phi_{t_2}) \\ A_{t_3}\, sin(\varepsilon_{t_3}\, t + \phi_{t_3}) \end{pmatrix} \text{ (c.f. section 7.2),}$$

$${}^{16}\boldsymbol{s}_{\hat{16},16} = |\boldsymbol{O}_{16} - \boldsymbol{O}_{\hat{16}}|,$$

$${}^{20}\boldsymbol{s}_{\hat{20},20} = |\boldsymbol{O}_{20} - \boldsymbol{O}_{\hat{20}}|.$$

E.4 Global projection matrix $\mathbf{P}$

The global projection matrix $\mathbf{P}$ for the mechanism (with the ship included) can be written as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_A & & & & & & & & & & \mathbf{0} \\ & \mathbf{P}_B & & & & & & & & & \\ & & \mathbf{P}_C & & & & & & & & \\ & & & \mathbf{P}_1 & & & & & & & \\ & & & & \mathbf{P}_2 & & & & & & \\ & & & & & \mathbf{P}_3 & & & & & \\ & & & & & & \mathbf{P}_4 & & & & \\ & & & & & & & \ddots & & & \\ & & & & & & & & & \mathbf{P}_{20} & \\ \mathbf{0} & & & & & & & & & & \mathbf{P}_{21} \end{bmatrix}$$

The global projection matrix $\mathbf{P}_{mc}$ for the mechanism only is the partition that lies within the dashed lines.

The projection matrices used to model the ship motion and stabilised platform are given by

$$\mathbf{P}_A = \begin{bmatrix} \mathbf{1} \\ \mathbf{O} \end{bmatrix}, \quad \mathbf{P}_B = \begin{bmatrix} \mathbf{O} \\ {}^{B}\mathbf{S}_A \end{bmatrix}, \quad \mathbf{P}_C = \begin{bmatrix} \mathbf{O} \\ {}^{C}\mathbf{S}_B \end{bmatrix}$$

where ${}^{B}\mathbf{S}_A$ and ${}^{C}\mathbf{S}_B$ are given in section 7.3.

The projection matrices used to model the one dof revolute joints are given by

$$\mathbf{P}_k = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \text{where k = 1,3,4,6,8,10.}$$

The projection matrices used to model the one dof prismatic joints are given by

$$\mathbf{P}_k = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \text{where k = 16, 20.}$$

The projection matrices used to model the three dof spherical joints positioned at O_2, O_5 and O_9 can be written as

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{O} \\ {}^{k}\mathbf{S}_{k-1} \end{bmatrix} \qquad \text{where k = 2,5,9}$$

where the 3-2-3 Euler projection matrices are given by

$${}^{k}\mathbf{S}_{k-1} = \begin{bmatrix} 0 & s\beta_{kc} & -c\beta_{kc}s\beta_{kb} \\ 0 & c\beta_{kc} & s\beta_{kb}s\beta_{kc} \\ 1 & 0 & c\beta_{kb} \end{bmatrix}.$$

The projection matrices used to model the two and three dof joints positioned at O_{12} and O_{13} can be written as

$$\mathbf{P}_{12} = \begin{bmatrix} \mathbf{O} \\ {}^{12}\mathbf{S}_{3} \end{bmatrix}$$

where the 3-2 Euler projection matrix is given by

$${}^{12}\mathbf{S}_{3} = \begin{bmatrix} 0 & -s\beta_{12b} \\ 1 & 0 \\ 0 & c\beta_{12b} \end{bmatrix}$$

and

$$\mathbf{P}_{13} = \begin{bmatrix} \mathbf{O} \\ {}^{13}\mathbf{S}_{12} \end{bmatrix}$$

where the 3-2-3 Euler projection matrix is given by

$${}^{13}\mathbf{S}_{12} = \begin{bmatrix} 0 & s\beta_{13c} & -c\beta_{13c}s\beta_{13b} \\ 0 & c\beta_{13c} & s\beta_{13b}s\beta_{13c} \\ 1 & 0 & c\beta_{13b} \end{bmatrix}.$$

The projection matrices used to model the two dof joints positioned at O_{15} and O_{19} can be written as

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{O} \\ {}^{k}\mathbf{S}_{12} \end{bmatrix} \qquad \text{where k = 15,19}$$

where the 3-2 Euler projection matrices are given by

$${}^{k}\mathbf{S}_{12} = \begin{bmatrix} 0 & -s\beta_{kb} \\ 1 & 0 \\ 0 & c\beta_{kb} \end{bmatrix}.$$

The projection matrices used to model the three dof spherical joints positioned at O_{17} and O_{21} can be written as

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{O} \\ {}^{k}\mathbf{S}_{k-1} \end{bmatrix} \qquad \text{where k = 17,21}$$

where the 3-2-1 Euler projection matrices are given by

$${}^{k}\mathbf{S}_{k-1} = \begin{bmatrix} 1 & 0 & -s\beta_{kb} \\ 0 & c\beta_{kc} & s\beta_{kc}c\beta_{kb} \\ 0 & -s\beta_{kc} & c\beta_{kc}c\beta_{kb} \end{bmatrix}.$$

E.5 The time derivative of the global interbody transformation matrix

The time derivative $\dot{\mathcal{J}}$ of the global interbody transformation matrix for the mechanism (with the ship included) is given by

$$\dot{\mathcal{J}} = -\mathcal{J}\dot{\mathcal{J}}^{-1}\mathcal{J}$$

where

$$\dot{\mathcal{J}}^{-1} = \left[\begin{array}{cccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-{}^{B}\dot{\mathcal{J}}_{A} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -{}^{C}\dot{\mathcal{J}}_{B} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -{}^{1}\dot{\mathcal{J}}_{C} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -{}^{2}\dot{\mathcal{J}}_{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -{}^{3}\dot{\mathcal{J}}_{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -{}^{4}\dot{\mathcal{J}}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -{}^{5}\dot{\mathcal{J}}_{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{6}\dot{\mathcal{J}}_{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -{}^{8}\dot{\mathcal{J}}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{9}\dot{\mathcal{J}}_{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{10}\dot{\mathcal{J}}_{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -{}^{12}\dot{\mathcal{J}}_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{13}\dot{\mathcal{J}}_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{15}\dot{\mathcal{J}}_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{16}\dot{\mathcal{J}}_{15} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{17}\dot{\mathcal{J}}_{16} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{19}\dot{\mathcal{J}}_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{20}\dot{\mathcal{J}}_{19} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -{}^{21}\dot{\mathcal{J}}_{20} & 0 & 0
\end{array}\right]$$

The time derivative $\dot{\mathcal{J}}^{-1}_{mc}$ of the global interbody transformation matrix for the mechanism only is the partition that lies within the dashed lines. The time derivatives ${}^{n+1}\dot{\mathcal{J}}_{n}$ of the interbody transformation matrices are given by

$${}^{n+1}\dot{\mathcal{J}}_{n} = {}^{n+1}\mathcal{G}_{n+1,n+1}\left({}^{n+1}\boldsymbol{v}^{\otimes}_{n+1,n+1}\right)^{T}\,{}^{n+1}\mathcal{G}^{-1}_{n+1,n+1}\,{}^{n+1}\mathcal{J}_{n}\,.$$

E.6 The time derivative of the global transformation matrix

The time derivative $\dot{\mathcal{G}}$ of the global transformation matrix for the mechanism (with the ship included) can be written as

$$\dot{\mathcal{G}} = \left[\begin{array}{ccccccc}
0 & & & & & & 0 \\
& 0 & & & & & \\
& & 0 & & & & \\
& & & 0 & & & \\
& & & & {}^{16}\dot{\mathcal{G}}_{16,16} & & \\
& & & & & 0 & \\
0 & & & & & & 0
\end{array}\right]$$

The time derivative $\dot{\mathcal{G}}_{mc}$ of the global transformation matrix for the mechanism only is the partition that lies within the dashed lines. The time derivatives ${}^{n+1}\dot{\mathcal{G}}_{n+1,n+1}$ of the transformation matrices are given by

$${}^{n+1}\dot{\mathcal{G}}_{n+1,n+1} = \begin{bmatrix} \mathbf{O} & -{}^{n+1}\mathbf{w}^{\times}_{n+1,n+1} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}.$$

E.7 The time derivative of the global projection matrix

The time derivative $\dot{\mathbf{P}}$ of the global projection matrix for the mechanism (with the ship included) can be written as

$$\dot{\mathbf{P}} = \begin{bmatrix}
\dot{\mathbf{P}}_A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \dot{\mathbf{P}}_B & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \dot{\mathbf{P}}_C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \dot{\mathbf{P}}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{15} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{17} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{19} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dot{\mathbf{P}}_{21}
\end{bmatrix}$$

The time derivative $\dot{\mathbf{P}}_{mc}$ of the global projection matrix for the mechanism only is the partition that lies within the dashed lines.

The time derivatives of the projection matrices used to model the ship motion and stabilised platform are given by

$$\dot{\mathbf{P}}_A = \begin{bmatrix} \mathbf{O} \\ \mathbf{O} \end{bmatrix}, \quad \dot{\mathbf{P}}_B = \begin{bmatrix} \mathbf{O} \\ {}^{B}\dot{\mathbf{S}}_A \end{bmatrix}, \quad \dot{\mathbf{P}}_C = \begin{bmatrix} \mathbf{O} \\ {}^{C}\dot{\mathbf{S}}_B \end{bmatrix}$$

where ${}^{B}\dot{\mathbf{S}}_A$ and ${}^{C}\dot{\mathbf{S}}_B$ are given in section 7.3.

The time derivatives of the projection matrices used to model the three dof spherical joints positioned at O_2, O_5 and O_9 can be written as

$$\dot{\mathbf{P}}_k = \begin{bmatrix} \mathbf{O} \\ {}^{k}\dot{\mathbf{S}}_{k-1} \end{bmatrix} \qquad \text{where k = 2,5,9}$$

where the time derivatives of the 3-2-3 Euler projection matrices are given by

$$ {}^{k}\mathbf{S}_{k-1} = \begin{bmatrix} 0 & \dot{\beta}_{kc} c\beta_{kc} & \dot{\beta}_{kc} s\beta_{kc} s\beta_{kb} - \dot{\beta}_{kb} c\beta_{kc} c\beta_{kb} \\ 0 & -\dot{\beta}_{kc} s\beta_{kc} & \dot{\beta}_{kb} c\beta_{kb} s\beta_{kc} + \dot{\beta}_{kc} s\beta_{kb} c\beta_{kc} \\ 0 & 0 & -\dot{\beta}_{kb} s\beta_{kb} \end{bmatrix}. $$

The time derivatives of the projection matrices used to model the two and three dof joints positioned at O_{12} and O_{13} can be written as

$$ \dot{\mathbf{P}}_{12} = \begin{bmatrix} \mathbf{O} \\ {}^{12}\dot{\mathbf{S}}_{3} \end{bmatrix} $$

where the time derivative of the 3-2 Euler projection matrix is given by

$$ {}^{12}\dot{\mathbf{S}}_{3} = \begin{bmatrix} 0 & -\dot{\beta}_{12b} c\beta_{12b} \\ 0 & 0 \\ 0 & -\dot{\beta}_{12b} s\beta_{12b} \end{bmatrix} $$

and

$$ \dot{\mathbf{P}}_{13} = \begin{bmatrix} \mathbf{O} \\ {}^{13}\dot{\mathbf{S}}_{12} \end{bmatrix} $$

where the time derivative of the 3-2-3 Euler projection matrix is given by

$$ {}^{13}\mathbf{S}_{12} = \begin{bmatrix} 0 & \dot{\beta}_{13c} c\beta_{13c} & \dot{\beta}_{13c} s\beta_{13c} s\beta_{13b} - \dot{\beta}_{13b} c\beta_{13c} c\beta_{13b} \\ 0 & -\dot{\beta}_{13} s\beta_{13c} & \dot{\beta}_{13b} c\beta_{13b} s\beta_{13c} + \dot{\beta}_{13c} s\beta_{13b} c\beta_{13c} \\ 0 & 0 & -\dot{\beta}_{13b} s\beta_{13b} \end{bmatrix}. $$

The time derivatives of the projection matrices used to model the two dof joints positioned at O_{15} and O_{19} can be written as

$$ \dot{\mathbf{P}}_{k} = \begin{bmatrix} \mathbf{O} \\ {}^{k}\dot{\mathbf{S}}_{12} \end{bmatrix} \qquad \text{where k = 15,19} $$

where the time derivatives of the 3-2 Euler projection matrices are given by

$$ {}^{k}\mathbf{S}_{12} = \begin{bmatrix} 0 & -\dot{\beta}_{kb} c\beta_{kb} \\ 0 & 0 \\ 0 & -\dot{\beta}_{kb} s\beta_{kb} \end{bmatrix}. $$

The time derivatives of the projection matrices used to model the three dof spherical joints positioned at O_{17} and O_{21} can be written as

$$ \dot{\mathbf{P}}_{k} = \begin{bmatrix} \mathbf{O} \\ {}^{k}\dot{\mathbf{S}}_{k-1} \end{bmatrix} \qquad \text{where k = 17,21} $$

where the time derivatives of the 3-2-1 Euler projection matrices are given by

$$ {}^{k}\mathbf{S}_{k-1} = \begin{bmatrix} 0 & 0 & -\dot{\beta}_{kb} c\beta_{kb} \\ 0 & -\dot{\beta}_{kc} s\beta_{kc} & \dot{\beta}_{kc} c\beta_{kc} c\beta_{kb} - \dot{\beta}_{kb} s\beta_{kc} s\beta_{kb} \\ 0 & -\dot{\beta}_{kc} c\beta_{kc} & -\dot{\beta}_{kc} s\beta_{kc} c\beta_{kb} - \dot{\beta}_{kb} c\beta_{kc} s\beta_{kb} \end{bmatrix}. $$

E.8 The global external force vector $\mathcal{F}_{Ext}$

The global external force vector for the mechanism (with the ship included) can be written as

$$\mathcal{F}_{Ext} = \begin{pmatrix} m_A{}^A g \\ {}^A c_A^{\times A} g \\ m_k{}^k g \\ {}^k c_k^{\times k} g \\ \vdots \\ \vdots \\ m_{21}{}^{21} g \\ {}^{21} c_{21}^{\times 21} g \end{pmatrix} \text{ where k = B,C,1,2,3,4,5,6,8,9,10,12,13,15,16,17,19,20.}$$

E.9 The global formulation of the vector $v^{\otimes}\mathbf{M}v$

The global formulation of the vector $v^{\otimes}\mathbf{M}v$ for the mechanism (with the ship included) can be written as

$$v^{\otimes}\mathbf{M}v = \begin{pmatrix} v_A^{\otimes A} M_A{}^A v_A \\ v_k^{\otimes k} M_k{}^k v_k \\ \vdots \\ v_{21}^{\otimes 21} M_{21}{}^{21} v_{21} \end{pmatrix} \text{ where k = B,C,1,2,3,4,5,6,8,9,10,12,13,15,16,17,19,20.}$$

E.10 Global mass matrix M

The global mass matrix $\mathbf{M}$ for the mechanism (with the ship included) can be written as

$$\mathbf{M} = \begin{bmatrix} {}^A M_A & & & & & & & & & & & & & & & & & & & 0 \\ & {}^B M_B & & & & & & & & & & & & & & & & & & \\ & & {}^C M_C & & & & & & & & & & & & & & & & & \\ & & & {}^1 M_1 & & & & & & & & & & & & & & & & \\ & & & & {}^2 M_2 & & & & & & & & & & & & & & & \\ & & & & & {}^3 M_3 & & & & & & & & & & & & & & \\ & & & & & & \ddots & & & & & & & & & & & & & \\ & & & & & & & {}^{20} M_{20} & & & & & & & & & & & & \\ & & & & & & & & {}^{21} M_{21} & & & & & & & & & & \\ 0 & & & & & & & & & & & & & & & & & & & \end{bmatrix}$$

The mass matrices ${}^n M_n$ of the links are not shown in detail. For the simulations carried out in Chapter 10 the arms of the mechanism were modelled as uniform thick walled tubes, the joints as point masses and the counterbalance masses as point masses.

Appendix F

Robot control code

The design of the control system has already been explained in Chapter 4. In this appendix the program used to control the two robots built at the University is presented.

F.1 Demkiwi1.c
This is the main module used to instruct the movement commands for the three dof robot.
F.2 Demcant1.c
This is the main module used to instruct the movement commands for the two dof robot.
F.3 Kiwi.h
This is the header file used for the kinematics module for the three dof robot.
F.4 Kiwi.kin
This is the kinematics module for the three dof robot.
F.5 Canttrk.h
This is the header file used for the kinematics module for the two dof robot.
F.6 Canttrk.kin
This is the kinematics module for the two dof robot.
F.7 Ph_ifx.h
This is the header file for the IFX driver module.
F.8 Ph_ifx.drv
This is the driver module for the IFX motor controller.
F.9 User.h
This is the header file for the users module.
F.10 User.c
This is the users module for the robot control programs.
F.11 Sysconfg.h
System configuration file for robot control programs.
F.12 Robot.h

F.1 Demkiwi1.c

```
/* DEMKIWI1.C demo file for robot control routines */

#include "ROBOT.H"
#include <alloc.h>
#include <string.h>
#include <math.h>
#include <STDIO.H>
#include <CONIO.H>
#include <DOS.H>
#include <STDLIB.H>
#include <CTYPE.H>
#include "PH_IFX.H"  /*should be removed*/

#include "user.h"
#include "kiwi.h"
#include "sysconfg.h"

#define DEBUG

void main()
{
   int RetCode = 0,s;
   char Ch,Str[100];
   SPACECOORD Pt,Orient,Centre;
   float Lamdai,Lamda,Radius,Height,Omega,Alpha3,Alpha2;

#if defined DEBUG
extern FILE *DebugFile;
#endif

   clrscr();

   switch (RetCode = Init_Robot())
   {
      case NULL :
         printf("\n\nInitialisation success.\n");
         break;

      case EIFXNOTREADY :
         do
         {
            printf("The IFX is not responding. (R)etry?\n");
            while(kbhit())
               getch();
            while(!kbhit());
            Ch = getch();
            if (toupper(Ch) != 'R')
               exit(RetCode);
         } while ((RetCode = Init_CheckIFX()) == EIFXNOTREADY);
         if (!(RetCode = Init_IFX()))
            break;

      default:
         printf("Initialisation error: %d\n",RetCode);
         exit(RetCode);
   }

//  Str = farmalloc(100*sizeof(CON_CODE));
   nosound();

   SetSpeed(2);
   Robot.way_dist = 10;   /* n degs */

   printf("Orbital tracking simulation for kiwibot.\n");
   printf("=======================================\n");
   printf("\nPress any key to home robot.");
   Wait;

   Home(0.5);

//  This only works for IFX controllers, reply does not come back until the move is
finished
   s = IDStat("1PR");
   GetStatus(s, Str, DOASK);
   s = IDStat("2PR");
   GetStatus(s, Str, DOASK);
   s = IDStat("3PR");
   GetStatus(s, Str, DOASK);

/*  Lamdai   - initial starting angular offset (degs)
    Lamda    - angle of rotation of tracked object (degs)
    Radius   - radius of circular path (m)
    Height   - distance of origin of circle from base (m)
    Omega    - angular velocity of tracked object (degs/s)
    Alpha3   - 3 angle (z axis) setting orientation of circle (degs)
    Alpha2   - 2 angle (y axis) (degs) */

//  Radius = 1000
//  Height = 577  30 deg El

   Lamdai = 0;
   Lamda = 360;
   Radius = 1000;
   Height = 1000;
   Omega = 10;
   Alpha3 = 0;
   Alpha2 = 0;

   PathtrkK(Lamdai,Lamda,Radius,Height,Omega,Alpha3,Alpha2);
   KiwiHoldPos(0.2);

   nosound();
   farfree(Str);
}
```

F.2 Demcant1.c

```
/* DEMCANT1.C demo file for Canterbury Tracker */

#include "ROBOT.H"
#include <alloc.h>
#include <string.h>
#include <math.h>
#include <STDIO.H>
#include <CONIO.H>
#include <DOS.H>
#include <STDLIB.H>
#include <CTYPE.H>
#include "PH_IFX.H"

#include "user.h"
#include "canttrk.h"
#include "sysconfg.h"

#define DEBUG

void main()
{
   int RetCode = 0,s;
   char Ch,Str[100];
   SPACECOORD Pt,Orient,Centre;
   float Lamdai,Lamda,Radius,Height,Omega,Alpha3,Alpha2;

#if defined DEBUG
extern FILE *DebugFile;
#endif

   clrscr();

   switch (RetCode = Init_Robot())
   {
      case NULL :
         printf("\n\nInitialisation success.\n");
         break;

      case EIFXNOTREADY :
         do
         {
            printf("The IFX is not responding. (R)etry?\n");
            while(kbhit())
            getch();
            while(!kbhit());
            Ch = getch();
            if (toupper(Ch) != 'R')
            exit(RetCode);
         } while ((RetCode = Init_CheckIFX()) == EIFXNOTREADY);
         if (!(RetCode = Init_IFX()))
            break;

      default:
         printf("Initialisation error: %d\n",RetCode);
         exit(RetCode);
   }

//  Str = farmalloc(100*sizeof(CON_CODE));
   nosound();
   SetSpeed(1);
   Robot.way_dist = 10;   /* n degs */

   printf("Orbital tracking simulation for Canterbury Tracker.\n");
   printf("===================================================\n");
   printf("\nPress any key to home robot.");
   Wait;

   Home(0.2);

   /* This only works for IFX controllers, reply does not come back until
   the move is finished */

   s = IDStat("1PR");
   GetStatus(s, Str, DOASK);
   s = IDStat("2PR");
   GetStatus(s, Str, DOASK);

/*  Lamdai   - initial starting angular offset (degs)
    Lamda    - angle of rotation of tracked object (degs)
    Radius-  - radius of circular path (m)
    Height-  - distance of origin of circle from base (m)
    Omega    -angular velocity of tracked object (degs/s)
    Alpha3   - 3 angle (z axis) setting orientation of circle (degs)
    Alpha2   - 2 angle (y axis) (degs) */
```

```
    Lamdai = 0;
    Lamda = 360;
    Radius = 1000;
    Height = 1000;   // 45deg
//  Height = 176.3;  // 10 deg
//  Height = 0;
//  Omega = 3;
    Omega = 15;
    Alpha3 = 0;
    Alpha2 = 0;

    PathtrkK(Lamdai,Lamda,Radius,Height,Omega,Alpha3,Alpha2);

    CantHoldPos(0.1);

    nosound();
    farfree(Str);
}
```

F.3 Kiwi.h

```
/*************************************************************************
*\
   KIWI.H
   Kinematics module header for the KIWI robot.
   ============================================

   This does not need to be explicitly included  in the robot control
   program, although it must be present or the program will not compile
   if #define KIWI is used in SYSCONFG.H.
   See ROBOT.H for more information.
\*************************************************************************
*/

/* Universal defines & typedefs */
/* Defines */
#define SPACEDIMEN      3           /* Number of coordinates necessary to
                                       define a point in the workspace */
#define MOTOR_NUMBER    3           /* Number of actuators */
#define MINSPEED        0.00001     /* Minimum endpoint speed */


#define HOME_ENDPT      {0,0,0}
#define MOTR_ENDPT      {0, 0, 0}
#define ROBOTSTRUCT_INIT { SPHERICAL,          \
                           VELO_MODE,          \
                           .02,                \
                           MAX_MOTOR_ACC,      \
                           0.02,               \
                           MOTR_ENDPT,         \
                           HOME_ENDPT          \
                         }

#undef  EBASE
#define EBASE           EBASEKIN
#define EBADPOSIT       EBASE + 1   /* Bad demand position sent to InvKin */
#define ENOTAVAIL       EBASE + 2   /* Function is not available */

/* insert error codes here */

/* Physical dimensions of Kiwibot */

//#define magOp   0.841
#define magOp   0.980

#define Phi1    0.0 * Pi/180        /* rotation about translated Z axis */
#define Theta1  0.0 * Pi/180        /* rotation about moved Y axis */
#define Psi1    0.0                 /* rotation about moved X axis */
#define a1      0.28033             /* displacement in X dir */
#define b1      0.0                 /* displacement in Y dir */
#define c1      0.0                 /* displacement in Z dir */
#define Phi2    120.0 * Pi/180
#define Theta2  0.0
#define Psi2    0.0*Pi/180
#define a2      -0.5*0.28033
#define b2      (sqrt(3.0)/2) * 0.28033
#define c2      0.0
#define Phi3    240*Pi/180
#define Theta3  0
#define Psi3    0
#define a3      -0.5*0.28033
#define b3      -(sqrt(3.0)/2) * 0.28033
#define c3      0

#define l1      0.841
#define l2      0.841
#define l3      0.841

#define nx1     cos(Phi1)*cos(Theta1)
#define ny1     sin(Phi1)*cos(Theta1)
#define nz1     -sin(Theta1)
#define sx1     cos(Phi1)*sin(Theta1)* sin(Psi1) - sin(Phi1)* cos(Psi1)
#define sy1     sin(Phi1)*sin(Theta1)* sin(Psi1) + cos(Phi1)* cos(Psi1)
#define sz1     cos(Theta1)* sin(Psi1)
#define ax1     cos(Phi1)* sin(Theta1)* cos(Psi1) + sin(Phi1)* sin(Psi1)
#define ay1     sin(Phi1)* sin(Theta1)* cos(Psi1) - cos(Phi1)* sin(Psi1)
#define az1     cos(Theta1)* cos(Psi1)
#define Px1     a1
#define Py1     b1
#define Pz1     c1

#define nx2     cos(Phi2)* cos(Theta2)
#define ny2     sin(Phi2)* cos(Theta2)
#define nz2     -sin(Theta2)
#define sx2     cos(Phi2)* sin(Theta2)* sin(Psi2) - sin(Phi2)* cos(Psi2)
#define sy2     sin(Phi2)* sin(Theta2)* sin(Psi2) + cos(Phi2)* cos(Psi2)
#define sz2     cos(Theta2)* sin(Psi2)
#define ax2     cos(Phi2)* sin(Theta2)* cos(Psi2) + sin(Phi2)* sin(Psi2)
#define ay2     sin(Phi2)* sin(Theta2)* cos(Psi2) - cos(Phi2)* sin(Psi2)
#define az2     cos(Theta2)* cos(Psi2)
#define Px2     a2
#define Py2     b2
#define Pz2     c2

#define nx3     cos(Phi3)* cos(Theta3)
#define ny3     sin(Phi3)* cos(Theta3)
#define nz3     -sin(Theta3)
#define sx3     cos(Phi3)* sin(Theta3)* sin(Psi3) - sin(Phi3)* cos(Psi3)
#define sy3     sin(Phi3)* sin(Theta3)* sin(Psi3) + cos(Phi3)* cos(Psi3)
#define sz3     cos(Theta3)* sin(Psi3)
#define ax3     cos(Phi3)* sin(Theta3)* cos(Psi3) + sin(Phi3)* sin(Psi3)
#define ay3     sin(Phi3)* sin(Theta3)* cos(Psi3) - cos(Phi3)* sin(Psi3)
#define az3     cos(Theta3)* cos(Psi3)
#define Px3     a3
#define Py3     b3
#define Pz3     c3

   /* Formulate trans matrices Tx1X,Tx2X,Tx3X */
#define Tx1X    {nx1,sx1,ax1,Px1;ny1,sy1,ay1,Py1;nz1,sz1,az1,Pz1;0,0,0,1}
#define Tx2X    {nx2,sx2,ax2,Px2;ny2,sy2,ay2,Py2;nz2,sz2,az2,Pz2;0,0,0,1}
#define Tx3X    {nx3,sx3,ax3,Px3,ny3,sy3,ay3,Py3;nz3,sz3,az3,Pz3;0,0,0,1}

/* Typedefs */
typedef float       SPACECOORD[SPACEDIMEN];   /* Space coordinate structure */
typedef float far * lpSPACECOORD;             /* Pointer to SPACECOORD */
typedef float       ACTCOORD[MOTOR_NUMBER];   /* Actuator coordinate structure */
typedef float far * lpACTCOORD;               /* Pointer to ACTCOORD */

/* Prototypes */
extern int Invkin(lpSPACECOORD, lpACTCOORD);
extern int Dirkin(lpACTCOORD, lpSPACECOORD);
int KinHome(float);
```

F.4 Kiwi.kin

```
/*************************************************************************\
   KIWI.KIN
   Kinematics module for the KIWI robot.
   =====================================

   This does not need to be explicitly included  in the robot control
   program, although it must be present or the program will not compile
   if #define KIWI is used in SYSCONFG.H.
   See ROBOT.H for more information.
\*************************************************************************/

#define KINEMATICS_LOADED

/* The following program is used to compute the inverse kinematics   */
/* for the KIWIBOT. Zero for arm angles (beta) taken from vertical */
/* Note - uses older vector notation (variables d etc) */

#include <stdio.h>
#include <math.h>
#include <alloc.h>
#include <conio.h>
#include "SYSCONFG.H"
#include "KIWI.H"
#include "USER.H"
#include <stdlib.h>
#include "ph_ifx.h"

int Invkin(lpSPACECOORD,lpACTCOORD);
void arm_end_pt_coords(int armNo,float Op[3],float Ba[4],float Bb[4]);
int Dirkin(lpACTCOORD,lpSPACECOORD);

void Dist_plane_pt(float P[3],float M[3],float Op[3],float *magd);
void Calcf(float Op[3],float M[3],float P[3],float y[3],float d[3],float *f);
void CalcS(float P[3],float f,float y[3],float d[3],float S[3]);
void Calc_Ball_jnt_postn(float l,float P[3],float S[3],float y[3],float d[3], float Ba[4],float
Bb[4]);
void arm_angles(int armNo,float Ba[4],float Bb[4],float *betaa,float *betab,float Baxyz[4],float
Bbxyz[4]);
void arm_angleprac(int armNo,float Baxyz[4],float Bbxyz[4],float betaa,float
betab,lpACTCOORD beta);

int Invkin(lpSPACECOORD theta,lpACTCOORD beta)
{
   int armNo;
   float   phi,xi,Ba[4],Baxyz[4],Bbxyz[4],Bb[4],betaa,betab,Op[3];
   phi = theta[2];
   xi = theta[1]/2;
```

```
    /* Top platform position */
    Op[0] = magOp*sin(xi)*cos(phi);
    Op[1] = magOp*sin(xi)*sin(phi);
    Op[2] = magOp*cos(xi);

    for(armNo=1;armNo<=3;armNo++)              /* arms 1 to 3 */
    {
        /* calc possible arm end pt coords */
        arm_end_pt_coords(armNo,Op,Ba,Bb);

        /* calc possible angles for arms */
        arm_angles(armNo,Ba,Bb,&betaa,&betab,Baxyz,Bbxyz);

        /* Routine to test which arm angle is used in practice */
        arm_angleprac(armNo,Baxyz,Bbxyz,betaa,betab,beta);
    }
    return(EXIT_SUCCESS);
}

void arm_end_pt_coords(int armNo,float Op[3],float Ba[4],float Bb[4])
{
    float P[3],y[3],M[3],l,half,magd,d[3],f,S[3];

    switch(armNo)
    {
        case 1:
            /* Arm origin positions in principal XYZ coord system */
            P[0] = a1;
            P[1] = b1;
            P[2] = c1;

            /* vector normals of circles described by arms */
            y[0] = sx1;
            y[1] = sy1;
            y[2] = sz1;

            /* Arm length */
            l = l1;

            break;

        case 2:
            /* Arm origin positions in principal XYZ coord system */
            P[0] = a2;
            P[1] = b2;
            P[2] = c2;

            /* vector normals of circles described by arms */
            y[0] = sx2;
            y[1] = sy2;
            y[2] = sz2;

            /* Arm length */
            l = l2;

            break;

        case 3:
            /* Arm origin positions in principal XYZ coord system */
            P[0] = a3;
            P[1] = b3;
            P[2] = c3;

            y[0] = sx3;
            y[1] = sy3;
            y[2] = sz3;

            /* Arm length */
            l = l3;

            break;

        default:
            break;
    }

    half = 0.5;
    MultScalVec(half,Op,M);

    /* distance between the points Pn and Qn (see thesis for defn) */
    /* Calculate magd = (P-M).Op/magOp */
    Dist_plane_pt(P,M,Op,&magd);

    /* vectors between points Pn and Qn (see thesis for defn) */
    /* Calculate the vector d */
    MultScalVec(magd,Op,d);

    /* calc fn (see thesis for defn) */
    /* Calculate the scalar quantity f */
    Calcf(Op,M,P,y,d,&f);

    /* calc Sn (see thesis for defn) */
    /* Calculate the scalar quantities s1 and s2 */
    CalcS(P,f,y,d,S);

    /* Calculate the two intersection points */
    Calc_Ball_jnt_postn(l,P,S,y,d,Ba,Bb);
}

/* Dirkin - converts a set of actuator coordinates to a set of spatial coordinates */
int Dirkin(lpACTCOORD Al, lpSPACECOORD Pt)
{
    /* Use AL & Pt to avert warning message */
    Al = Al;
    Pt = Pt;
    return(ENOTAVAIL);
}

/* KinHome - performs any movements necessary to make the robot ready
for safe homing procedure */
int KinHome(float andy)
{
    return(EXIT_SUCCESS);
}


void Dist_plane_pt(float P[3],float M[3],float Op[3],float *magd)
{
    /* magd*/
    float scal1,vec1[3];

    SubtractVector(M,P,vec1);
    DotProdVector(Op,vec1,&scal1);
    *magd = fabs(scal1/magOp);
}

void Calcf(float Op[3],float M[3],float P[3],float y[3],float d[3],float *f)
{
 /* f = */

 float scal1,scal2,scal3,scal4,vec1[3],vec2[3];

    DotProdVector(Op,M,&scal1);
    DotProdVector(Op,P,&scal2);
    scal3 = scal1 - scal2;
    CrossprodVec(y,d,vec1);
    CrossprodVec(vec1,y,vec2);
    DotProdVector(Op,vec2,&scal4);
    *f = scal3/scal4;
}

void CalcS(float P[3],float f,float y[3],float d[3],float S[3])
{
    float vec1[3],vec2[3],vec3[3];
    CrossprodVec(y,d,vec1);
    CrossprodVec(vec1,y,vec2);
    MultScalVec(f,vec2,vec3);
    AddVector(P,vec3,S);
}

void Calc_Ball_jnt_postn(float l,float P[3],float S[3],float y[3],float d[3],float Ba[4],float Bb[4])
{
    float a,b,t,vec1[3],vec2[3],yd[3];

    /* Calc yxd */
    CrossprodVec(y,d,yd);

    /* calculate t (see thesis for defn) */

    a =     P[0]*yd[0] - S[0]*yd[0] + P[1]*yd[1]
          - S[1]*yd[1] + P[2]*yd[2] - S[2]*yd[2];

    b =   -(pow(P[1],2)*pow(yd[0],2)) - pow(P[2],2)*pow(yd[0],2)
          + pow(l,2)*pow(yd[0],2) + 2*P[1]*S[1]*pow(yd[0],2)
          - pow(S[1],2)*pow(yd[0],2) + 2*P[2]*S[2]*pow(yd[0],2)
          - pow(S[2],2)*pow(yd[0],2) + 2*P[0]*P[1]*yd[0]*yd[1]
          - 2*P[1]*S[0]*yd[0]*yd[1]- 2*P[0]*S[1]*yd[0]*yd[1]
          + 2*S[0]*S[1]*yd[0]*yd[1] - pow(P[0],2)*pow(yd[1],2)
          - pow(P[2],2)*pow(yd[1],2) + pow(l,2)*pow(yd[1],2)
          + 2*P[0]*S[0]*pow(yd[1],2) - pow(S[0],2)*pow(yd[1],2)
          + 2*P[2]*S[2]*pow(yd[1],2) - pow(S[2],2)*pow(yd[1],2)
          + 2*P[0]*P[2]*yd[0]*yd[2] - 2*P[2]*S[0]*yd[0]*yd[2]
          - 2*P[0]*S[2]*yd[0]*yd[2] + 2*S[0]*S[2]*yd[0]*yd[2]
          + 2*P[1]*P[2]*yd[1]*yd[2] - 2*P[2]*S[1]*yd[1]*yd[2]
          - 2*P[1]*S[2]*yd[1]*yd[2] + 2*S[1]*S[2]*yd[1]*yd[2]
          - pow(P[0],2)*pow(yd[2],2)  - pow(P[1],2)*pow(yd[2],2)
          + pow(l,2)*pow(yd[2],2) + 2*P[0]*S[0]*pow(yd[2],2)
          - pow(S[0],2)*pow(yd[2],2) + 2*P[1]*S[1]*pow(yd[2],2)
          - pow(S[1],2)*pow(yd[2],2);

    t = pow(a+b,0.5)/(pow(yd[0],2) + pow(yd[1],2) + pow(yd[2],2));

    /* arm end point positions */
    MultScalVec(t,yd,vec1);
    AddVector(S,vec1,Ba);
    SubtractVector(vec1,S,Bb);
    Ba[3] = 1.0;
    Bb[3] = 1.0;
}
```

```
void arm_angles(int armNo,float Ba[4],float Bb[4],float *betaa,float *betab,float
Baxyz[4],float Bbxyz[4])
{
    float TXx1[4][4],TXx2[4][4],TXx3[4][4];

    TXx1[0][0] = nx1;
    TXx1[0][1] = ny1;
    TXx1[0][2] = nz1;
    TXx1[0][3] = -(nx1*Px1+ny1*Py1+nz1*Pz1);
    TXx1[1][0] = sx1;
    TXx1[1][1] = sy1;
    TXx1[1][2] = sz1;
    TXx1[1][3] = -(sx1*Px1+sy1*Py1+sz1*Pz1);
    TXx1[2][0] = ax1;
    TXx1[2][1] = ay1;
    TXx1[2][2] = az1;
    TXx1[2][3] = -(ax1*Px1+ay1*Py1+az1*Pz1);
    TXx1[3][0] = 0;
    TXx1[3][1] = 0;
    TXx1[3][2] = 0;
    TXx1[3][3] = 1;

    TXx2[0][0] = nx2;
    TXx2[0][1] = ny2;
    TXx2[0][2] = nz2;
    TXx2[0][3] = -(nx2*Px2+ny2*Py2+nz2*Pz2);
    TXx2[1][0] = sx2;
    TXx2[1][1] = sy2;
    TXx2[1][2] = sz2;
    TXx2[1][3] = -(sx2*Px2+sy2*Py2+sz2*Pz2);
    TXx2[2][0] = ax2;
    TXx2[2][1] = ay2;
    TXx2[2][2] = az2;
    TXx2[2][3] = -(ax2*Px2+ay2*Py2+az2*Pz2);
    TXx2[3][0] = 0;
    TXx2[3][1] = 0;
    TXx2[3][2] = 0;
    TXx2[3][3] = 1;

    TXx3[0][0] = nx3;
    TXx3[0][1] = ny3;
    TXx3[0][2] = nz3;
    TXx3[0][3] = -(nx3*Px3+ny3*Py3+nz3*Pz3);
    TXx3[1][0] = sx3;
    TXx3[1][1] = sy3;
    TXx3[1][2] = sz3;
    TXx3[1][3] = -(sx3*Px3+sy3*Py3+sz3*Pz3);
    TXx3[2][0] = ax3;
    TXx3[2][1] = ay3;
    TXx3[2][2] = az3;
    TXx3[2][3] = -(ax3*Px3+ay3*Py3+az3*Pz3);
    TXx3[3][0] = 0;
    TXx3[3][1] = 0;
    TXx3[3][2] = 0;
    TXx3[3][3] = 1;

    /* The points Ba & Bb are expressed in the principal coordinate frame */
    switch(armNo)
    {
        case 1:
            MultMatVec4(TXx1,Ba,Baxyz);
            MultMatVec4(TXx1,Bb,Bbxyz);
            break;

        case 2:
            MultMatVec4(TXx2,Ba,Baxyz);
            MultMatVec4(TXx2,Bb,Bbxyz);
            break;

        case 3:
            MultMatVec4(TXx3,Ba,Baxyz);
            MultMatVec4(TXx3,Bb,Bbxyz);
            break;

        default:
            break;
    }

    *betaa = acos((Baxyz[2])/
            (sqrt(Baxyz[0]*Baxyz[0] + Baxyz[1]*Baxyz[1] + Baxyz[2]*Baxyz[2])));

    *betab = acos((Bbxyz[2])/
            (sqrt(Bbxyz[0]*Bbxyz[0] + Bbxyz[1]*Bbxyz[1] + Bbxyz[2]*Bbxyz[2])));
}

/* Routine to test which arm angle is used in practice */
void arm_angleprac(int armNo,float Baxyz[4],float Bbxyz[4],float betaa,float
betab,lpACTCOORD beta)
{
    /* check if angles negative */
    if (Baxyz[0] < 0)
        betaa = -betaa;
    else
        betaa = betaa;

    if (Bbxyz[0] < 0)
        betab = -betab;
    else
        betab = betab;

    /* choose appropriate angle */
    if (betaa > betab)
        beta[armNo-1] = betaa;
    else
        beta[armNo-1] = betab;
}
```

F.5 Canttrk.h

```
/*****************************************************************\
    CANTTRK.H
    Kinematics module header for the Canterbury Tracker robot.
    ==========================================================

    This does not need to be explicitly included  in the robot control
    program, although it must be present or the program will not compile
    if #define KIWI is used in SYSCONFG.H.
    See ROBOT.H for more information.
\*****************************************************************/

/* Universal defines & typedefs */
/* Defines */
#define SPACEDIMEN 3            /* Number of coordinates necessary to
                                   define a point in the workspace */
#define MOTOR_NUMBER  3         /* Number of actuators */
#define MINSPEED    0.00001     /* Minimum endpoint speed */

#define HOME_ENDPT    {0,0,0}
#define MOTR_ENDPT {0, 0, 0}
#define ROBOTSTRUCT_INIT  { SPHERICAL,        \
                            VELO_MODE,        \
                            .02,              \
                            MAX_MOTOR_ACC,    \
                            0.02,             \
                            MOTR_ENDPT,       \
                            HOME_ENDPT        \
                            }

#undef  EBASE
#define EBASE           EBASEKIN
#define EBADPOSIT       EBASE + 1     /* Bad demand position sent to InvKin */
#define ENOTAVAIL       EBASE + 2     /* Function is not available */

/* insert error codes here */

/* Physical dimensions of Kiwibot */

//#define magOp 0.841
#define magOp 0.900

#define Phi1    0.0 * Pi/180        /* rotation about translated Z axis */
#define Theta1  0.0 * Pi/180        /* rotation about moved Y axis */
#define Psi1    0.0                 /* rotation about moved X axis */
#define a1      0.28033             /* displacement in X dir */
#define b1      0.0                 /* displacement in Y dir */
#define c1      0.0                 /* displacement in Z dir */
#define Phi2    120.0 * Pi/180
#define Theta2  0.0
#define Psi2    0.0*Pi/180
#define a2      -0.5*0.28033
#define b2      (sqrt(3.0)/2) * 0.28033
#define c2      0.0
#define Phi3    240*Pi/180
#define Theta3  0
#define Psi3    0
#define a3      -0.5*0.28033
#define b3      -(sqrt(3.0)/2) * 0.28033
#define c3      0

#define l1      0.841
#define l2      0.841
#define l3      0.841

#define nx1     cos(Phi1)*cos(Theta1)
#define ny1     sin(Phi1)*cos(Theta1)
#define nz1     -sin(Theta1)
#define sx1     cos(Phi1)*sin(Theta1)* sin(Psi1) - sin(Phi1)* cos(Psi1)
#define sy1     sin(Phi1)*sin(Theta1)* sin(Psi1) + cos(Phi1)* cos(Psi1)
#define sz1     cos(Theta1)* sin(Psi1)
#define ax1     cos(Phi1)* sin(Theta1)* cos(Psi1) + sin(Phi1)* sin(Psi1)
#define ay1     sin(Phi1)* sin(Theta1)* cos(Psi1) - cos(Phi1)* sin(Psi1)
#define az1     cos(Theta1)* cos(Psi1)
#define Px1     a1
#define Py1     b1
#define Pz1     c1

#define nx2     cos(Phi2)* cos(Theta2)
#define ny2     sin(Phi2)* cos(Theta2)
#define nz2     -sin(Theta2)
#define sx2     cos(Phi2)* sin(Theta2)* sin(Psi2) - sin(Phi2)* cos(Psi2)
#define sy2     sin(Phi2)* sin(Theta2)* sin(Psi2) + cos(Phi2)* cos(Psi2)
#define sz2     cos(Theta2)* sin(Psi2)
#define ax2     cos(Phi2)* sin(Theta2)* cos(Psi2) + sin(Phi2)* sin(Psi2)
```

```
#define   ay2       sin(Phi2)* sin(Theta2)* cos(Psi2) - cos(Phi2)* sin(Psi2)
#define   az2       cos(Theta2)* cos(Psi2)
#define   Px2       a2
#define   Py2       b2
#define   Pz2       c2

#define   nx3       cos(Phi3)* cos(Theta3)
#define   ny3       sin(Phi3)* cos(Theta3)
#define   nz3       -sin(Theta3)
#define   sx3       cos(Phi3)* sin(Theta3)* sin(Psi3) - sin(Phi3)* cos(Psi3)
#define   sy3       sin(Phi3)* sin(Theta3)* sin(Psi3) + cos(Phi3)* cos(Psi3)
#define   sz3       cos(Theta3)* sin(Psi3)
#define   ax3       cos(Phi3)* sin(Theta3)* cos(Psi3) + sin(Phi3)* sin(Psi3)
#define   ay3       sin(Phi3)* sin(Theta3)* cos(Psi3) - cos(Phi3)* sin(Psi3)
#define   az3       cos(Theta3)* cos(Psi3)
#define   Px3       a3
#define   Py3       b3
#define   Pz3       c3

/* Formulate trans matrices Tx1X,Tx2X,Tx3X */
#define   Tx1X      {nx1,sx1,ax1,Px1;ny1,sy1,ay1,Py1;nz1,sz1,az1,Pz1;0,0,0,1}
#define   Tx2X      {nx2,sx2,ax2,Px2;ny2,sy2,ay2,Py2;nz2,sz2,az2,Pz2;0,0,0,1}
#define   Tx3X      {nx3,sx3,ax3,Px3;ny3,sy3,ay3,Py3;nz3,sz3,az3,Pz3;0,0,0,1}

/* Typedefs */
typedef float       SPACECOORD[SPACEDIMEN];     /* Space coordinate
structure */
typedef float far * lpSPACECOORD;               /* Pointer to
SPACECOORD */
typedef float       ACTCOORD[MOTOR_NUMBER];     /* Actuator coordinate
structure */
typedef float far * lpACTCOORD;                 /* Pointer to
ACTCOORD */

/* Prototypes */
extern int Invkin(lpSPACECOORD, lpACTCOORD);
extern int Dirkin(lpACTCOORD, lpSPACECOORD);
int KinHome(float);
```

F.6 Canttrk.kin

```
/*****************************************************************
*\
   CANTTRK.KIN
   Kinematics module for the Canterbury tracker
   ============================================
   This does not need to be explicitly included  in the robot control
   program, although it must be present or the program will not compile
   if #define KIWI is used in SYSCONFG.H.
   See ROBOT.H for more information.
\*****************************************************************
*/

#define KINEMATICS_LOADED

/* The following program is used to compute the inverse kinematics   */
/* for the KIWIBOT. Zero for arm angles (beta) taken from vertical */
/* Note - uses older vector notation (variables d etc) */

#include <stdio.h>
#include <math.h>
#include <alloc.h>
#include <conio.h>
#include "SYSCONFG.H"
#include "CANTTRK.H"
#include "USER.H"
#include <stdlib.h>
#include "ph_ifx.h"

int Invkin(lpSPACECOORD,lpACTCOORD);
void arm_end_pt_coords(int armNo,float Op[3],float Ba[4],float Bb[4]);
int Dirkin(lpACTCOORD,lpSPACECOORD);

void Dist_plane_pt(float P[3],float M[3],float Op[3],float *magd);
void Calcf(float Op[3],float M[3],float P[3],float y[3],float d[3],float *f);
void CalcS(float P[3],float f,float y[3],float d[3],float S[3]);
void Calc_Ball_jnt_postn(float l,float P[3],float S[3],float y[3],float d[3],float Ba[4],float
Bb[4]);
void arm_angles(int armNo,float Ba[4],float Bb[4],float *betaa,float *betab,float
Baxyz[4],float Bbxyz[4]);
void arm_angleprac(int armNo,float Baxyz[4],float Bbxyz[4],float betaa,float
betab,lpACTCOORD beta);

int Invkin(lpSPACECOORD theta,lpACTCOORD beta)
{
   int armNo;
   float   phi,xi,Ba[4],Baxyz[4],Bbxyz[4],Bb[4],betaa,betab,Op[3];
   phi = theta[2];
   xi = theta[1]/2;

   /* Top platform position */
   Op[0] = magOp*sin(xi)*cos(phi);
   Op[1] = magOp*sin(xi)*sin(phi);
   Op[2] = magOp*cos(xi);

   for(armNo=1;armNo<=2;armNo++)              /* arms 1 to 3 */
   {
      /* calc possible arm end pt coords */
      arm_end_pt_coords(armNo,Op,Ba,Bb);

      /* calc possible angles for arms */
      arm_angles(armNo,Ba,Bb,&betaa,&betab,Baxyz,Bbxyz);

      /* Routine to test which arm angle is used in practice */
      arm_angleprac(armNo,Baxyz,Bbxyz,betaa,betab,beta);
   }

   beta[2] =0;    // Set 3rd arm to 0 deg

   return(EXIT_SUCCESS);
}

void arm_end_pt_coords(int armNo,float Op[3],float Ba[4],float Bb[4])
{
   float P[3],y[3],M[3],l,half,magd,d[3],f,S[3];

   switch(armNo)
   {
      case 1:
         /* Arm origin positions in principal XYZ coord system */
         P[0] = a1;
         P[1] = b1;
         P[2] = c1;

         /* vector normals of circles described by arms */
         y[0] = sx1;
         y[1] = sy1;
         y[2] = sz1;

         /* Arm length */
         l = l1;

         break;

      case 2:
         /* Arm origin positions in principal XYZ coord system */
         P[0] = a2;
         P[1] = b2;
         P[2] = c2;

         /* vector normals of circles described by arms */
         y[0] = sx2;
         y[1] = sy2;
         y[2] = sz2;

         /* Arm length */
         l = l2;

         break;

      default:
         break;
   }

   half = 0.5;
   MultScalVec(half,Op,M);

   /* distance between the points Pn and Qn (see thesis for defn) */
   /* Calculate magd = (P-M).Op/magOp */
   Dist_plane_pt(P,M,Op,&magd);

   /* vectors between points Pn and Qn (see thesis for defn) */
   /* Calculate the vector d */
   MultScalVec(magd,Op,d);

   /* calc fn (see thesis for defn) */
   /* Calculate the scalar quantity f */
   Calcf(Op,M,P,y,d,&f);

   /* calc Sn (see thesis for defn) */
   /* Calculate the scalar quantities s1 and s2 */
   CalcS(P,f,y,d,S);

   /* Calculate the two intersection points */
   Calc_Ball_jnt_postn(l,P,S,y,d,Ba,Bb);
}

/* Dirkin - converts a set of actuator coordinates to a set of spatial coordinates. */
int Dirkin(lpACTCOORD Al, lpSPACECOORD Pt)
{
   /* Use AL & Pt to avert warning message */
   Al = Al;
   Pt = Pt;

   return(ENOTAVAIL);
}

/* KinHome - performs any movements necessary to make the robot ready for safe homing
procedure */
int KinHome(float andy)
{
   return(EXIT_SUCCESS);
}
```

```
void Dist_plane_pt(float P[3],float M[3],float Op[3],float *magd)
{
    /* magd*/
    float scal1,vec1[3];
    SubtractVector(M,P,vec1);
    DotProdVector(Op,vec1,&scal1);
    *magd = fabs(scal1/magOp);
}

void Calcf(float Op[3],float M[3],float P[3],float y[3],float d[3],float *f)
{
    /* f */
    float scal1,scal2,scal3,scal4,vec1[3],vec2[3];
    DotProdVector(Op,M,&scal1);
    DotProdVector(Op,P,&scal2);
    scal3 = scal1 - scal2;
    CrossprodVec(y,d,vec1);
    CrossprodVec(vec1,y,vec2);
    DotProdVector(Op,vec2,&scal4);
    *f = scal3/scal4;
}

void CalcS(float P[3],float f,float y[3],float d[3],float S[3])
{
    float vec1[3],vec2[3],vec3[3];
    CrossprodVec(y,d,vec1);
    CrossprodVec(vec1,y,vec2);
    MultScalVec(f,vec2,vec3);
    AddVector(P,vec3,S);
}

void Calc_Ball_jnt_postn(float l,float P[3],float S[3],float y[3],float d[3], float
Ba[4],float Bb[4])
{
    float a,b,t,vec1[3],vec2[3],yd[3];
    /* Calc yxd */
    CrossprodVec(y,d,yd);

    /* calculate t (see thesis for defn) */
    a =     P[0]*yd[0] - S[0]*yd[0] + P[1]*yd[1]
          - S[1]*yd[1] + P[2]*yd[2] - S[2]*yd[2];

    b =   -(pow(P[1],2)*pow(yd[0],2)) - pow(P[2],2)*pow(yd[0],2)
          + pow(l,2)*pow(yd[0],2) + 2*P[1]*S[1]*pow(yd[0],2)
          - pow(S[1],2)*pow(yd[0],2) + 2*P[2]*S[2]*pow(yd[0],2)
          - pow(S[2],2)*pow(yd[0],2) + 2*P[0]*P[1]*yd[0]*yd[1]
          - 2*P[1]*S[0]*yd[0]*yd[1]- 2*P[0]*S[1]*yd[0]*yd[1]
          + 2*S[0]*S[1]*yd[0]*yd[1] - pow(P[0],2)*pow(yd[1],2)
          - pow(P[2],2)*pow(yd[1],2) + pow(l,2)*pow(yd[1],2)
          + 2*P[0]*S[0]*pow(yd[1],2) - pow(S[0],2)*pow(yd[1],2)
          + 2*P[2]*S[2]*pow(yd[1],2) - pow(S[2],2)*pow(yd[1],2)
          + 2*P[0]*P[2]*yd[0]*yd[2] - 2*P[2]*S[0]*yd[0]*yd[2]
          - 2*P[0]*S[2]*yd[0]*yd[2] + 2*S[0]*S[2]*yd[0]*yd[2]
          + 2*P[1]*P[2]*yd[1]*yd[2] - 2*P[2]*S[1]*yd[1]*yd[2]
          - 2*P[1]*S[2]*yd[1]*yd[2] + 2*S[1]*S[2]*yd[1]*yd[2]
          - pow(P[0],2)*pow(yd[2],2) - pow(P[1],2)*pow(yd[2],2)
          + pow(l,2)*pow(yd[2],2) + 2*P[0]*S[0]*pow(yd[2],2)
          - pow(S[0],2)*pow(yd[2],2) + 2*P[1]*S[1]*pow(yd[2],2)
          - pow(S[1],2)*pow(yd[2],2);

    t = pow(a+b,0.5)/(pow(yd[0],2) + pow(yd[1],2) + pow(yd[2],2));

    /* arm end point positions */
    MultScalVec(t,yd,vec1);
    AddVector(S,vec1,Ba);
    SubtractVector(vec1,S,Bb);
    Ba[3] = 1.0;
    Bb[3] = 1.0;
}

void arm_angles(int armNo,float Ba[4],float Bb[4],float *betaa,float *betab,float
Baxyz[4],float Bbxyz[4])
{
    float TXx1[4][4],TXx2[4][4];
    TXx1[0][0] = nx1;
    TXx1[0][1] = ny1;
    TXx1[0][2] = nz1;
    TXx1[0][3] = -(nx1*Px1+ny1*Py1+nz1*Pz1);
    TXx1[1][0] = sx1;
    TXx1[1][1] = sy1;
    TXx1[1][2] = sz1;
    TXx1[1][3] = -(sx1*Px1+sy1*Py1+sz1*Pz1);
    TXx1[2][0] = ax1;
    TXx1[2][1] = ay1;
    TXx1[2][2] = az1;
    TXx1[2][3] = -(ax1*Px1+ay1*Py1+az1*Pz1);
    TXx1[3][0] = 0;
    TXx1[3][1] = 0;
    TXx1[3][2] = 0;
    TXx1[3][3] = 1;

    TXx2[0][0] = nx2;
    TXx2[0][1] = ny2;
    TXx2[0][2] = nz2;
    TXx2[0][3] = -(nx2*Px2+ny2*Py2+nz2*Pz2);
    TXx2[1][0] = sx2;
    TXx2[1][1] = sy2;
    TXx2[1][2] = sz2;
    TXx2[1][3] = -(sx2*Px2+sy2*Py2+sz2*Pz2);
    TXx2[2][0] = ax2;
    TXx2[2][1] = ay2;
    TXx2[2][2] = az2;
    TXx2[2][3] = -(ax2*Px2+ay2*Py2+az2*Pz2);
    TXx2[3][0] = 0;
    TXx2[3][1] = 0;
    TXx2[3][2] = 0;
    TXx2[3][3] = 1;

    /* The points Ba & Bb are expressed in the principal coordinate frame */
    switch(armNo)
    {
        case 1:
            MultMatVec4(TXx1,Ba,Baxyz);
            MultMatVec4(TXx1,Bb,Bbxyz);
            break;

        case 2:
            MultMatVec4(TXx2,Ba,Baxyz);
            MultMatVec4(TXx2,Bb,Bbxyz);
            break;

     default:
            break;
    }

  *betaa = acos((Baxyz[2])/
            (sqrt(Baxyz[0]*Baxyz[0] + Baxyz[1]*Baxyz[1] + Baxyz[2]*Baxyz[2])));

  *betab = acos((Bbxyz[2])/
            (sqrt(Bbxyz[0]*Bbxyz[0] + Bbxyz[1]*Bbxyz[1] + Bbxyz[2]*Bbxyz[2])));
}

/* Routine to test which arm angle is used in practice */
void    arm_anglepract(int    armNo,float    Baxyz[4],float    Bbxyz[4],float    betaa,float
betab,lpACTCOORD beta)
{
    /* check if angles negative */
    if (Baxyz[0] < 0)
        betaa = -betaa;
    else
        betaa = betaa;

    if (Bbxyz[0] < 0)
        betab = -betab;
    else
        betab = betab;

    /* choose appropriate angle */
    if (betaa > betab)
        beta[armNo-1] = betaa;
    else
        beta[armNo-1] = betab;
}
```

F.7 Ph_ifx.h

```
/*************************************************************************\
    PH_IFX.H
    Motor driver header for robot control programs.
    ================================================

    PARKER-HANNIFIN Digiplan IFX motor driver.

    This does not need to be explicitly included  in the robot control
    program. It should be specified in the Motor Driver section of
    SYSCONFG.H. In addition the variable BASEPORT should be specified
    as either COM1 or COM2 depending on the serial port that the IFX
    is attached to.
    Other parameters that should be included in SYSCONFG.H are:
        EBASEDRIVER             -start of driver error codes.
        GEAR_RATIO              -reduction ratio of speed reducers.
        MOTOR_RES               -step per rev. resolution of motors.
        SLOW,MEDIUM,FAST        -standard homing speeds (rev/sec).
        MAX_HOME_RATE           -maximum homing speed (rev/sec).
        MAX_MOTOR_VEL           -maximum allowable motor speed (rev/sec).
        MAX_MOTOR_ACC           -maximum allowable motor acceleration (rev/secý)
        MOTOR_OFFSETS           -Offset from home position to kinematic datum
                                 in motor steps (1 =< n =< 8). Define as:
        #define MOTOR_OFFSETS {nnnn,nnnn,nnnn, -- ,nnnn}

    Notes
    > The IFX status command nW1 has not been implemented in this
    driver because it breaks the general syntax rules for X code. Using
    it may adversely affect the data transmission process.
\*************************************************************************/

#include "SYSCONFG.H"

/*********************************************************\
    Universal defines & typedefs (all drivers have these)
\*********************************************************/
typedef unsigned char       CON_CODE;
typedef unsigned char far * lpCON_CODE;
```

```
/**********************************\
   Parker Hannifin IFX constants
\**********************************/
#define IFX_VERSION             "*92-7818-01A5"
#define EMERG_STOP_STRING       " K K K K K K K K "
#define IFXTIMEOUT              1                   /* Wait time (s) for IFX
to reply on init. */
#define XCODE_MAXLEN            15                  /* Max. length of
XCODE reply or command */
#define MAX_STATUS              20                  /* Number of status
commands supported */
#define MAX_MOTOR_NUMBER        8                   /* Max. number of motors
that IFX supports */
#define DELIMITERS              " \r"               /* Xcode delimiters
(Space)(Return) */
#define CTRLCODE_LENGTH         200                 /* Length of Xcode string
(used in _Move) */

/**********************************\
   Buffer Maintainance Routines
\**********************************/
/* Error Codes */
#undef EBASE
#define EBASE           EBASEDRIVER
#define EBUFFUL         EBASEDRIVER + 1     /* Buffer is full */
#define EBUFEMPTY       EBASEDRIVER + 2     /* Buffer is empty */
#define EINITCOM        EBASEDRIVER + 3     /* Communications initialisation
error */
#define ESENDBUF        EBASEDRIVER + 4     /* Can't
ReadBuffer(&C,SEND_BUFFER) */
#define ESENDBUFMT      EBASEDRIVER + 5     /* Transmit while send buffer is
empty */
#define ERECVBUFFUL     EBASEDRIVER + 6     /* Trying to recv while recv buf
full */
#define EBADCHAR        EBASEDRIVER + 7     /* Attempt to send non Xcode
character */
#define EIFXNOTREADY    EBASEDRIVER + 8     /* IFX controller not responding
*/
#define EIFXBADREPLY    EBASEDRIVER + 9     /* IFX echoes incorrectly at init
*/
#define ENOTSTATUS      EBASEDRIVER + 10    /* Not a valid status structure
index */
#define ETOOFAST        EBASEDRIVER + 12    /* Max motor velocity exceeded
*/
#define ENOENABLE       EBASEDRIVER + 13    /* Not able to enable movement
*/

/* Buffer control codes (define status codes last) */
#define BC_BASE         200
#define BC_STAT_BASE    201             /* Base of status command BC codes */
#define BC_PAUSE        BC_BASE + 0     /* Pause sending data */
#define BC_B            BC_BASE + 1     /* 'B' Xcode status command prefix */
#define BC_BS           BC_BASE + 2     /* 'BS' */
#define BC_IS           BC_BASE + 3     /* 'IS' */
#define BC_R            BC_BASE + 4     /* 'R' */
#define BC_RA           BC_BASE + 5     /* 'RA' */
#define BC_RB           BC_BASE + 6     /* 'RB' */
#define BC_RC           BC_BASE + 7     /* 'RC' */
#define BC_RS           BC_BASE + 8     /* 'RS' */
#define BC_TS           BC_BASE + 9     /* 'TS' */
#define BC_W3           BC_BASE +10     /* 'W3' */
#define BC_XSR          BC_BASE +11     /* 'XSR' */
#define BC_FR           BC_BASE +12     /* 'FR' */
#define BC_PR           BC_BASE +13     /* 'PR' */
#define BC_PX           BC_BASE +14     /* 'PX' */
#define BC_RV           BC_BASE +15     /* 'RV' */
#define BC_STM          BC_BASE +16     /* 'STM' */
#define BC_XC           BC_BASE +17     /* 'XC' */
#define BC_XSD          BC_BASE +18     /* 'XSD' */
#define BC_XSP          BC_BASE +19     /* 'XSP' */
#define BC_XSS          BC_BASE +20     /* 'XSS' */
#define NO_STAT         BC_BASE +21     /* No status commands are pending */

/* Defines & typedefs */
#define SEND_BUFFER_SIZE 500
#define RISETIME        MAX_MOTOR_VEL/MAX_MOTOR_ACC

/* DOASK & NOASK are GetStatus bit params */
#define DOASK       1       /* Ask controller for status information */
#define NOASK       2       /* Do not ask controller for status information */
#define NOWAIT      4
   /* EnableM,GetStatus bit param. Do not wait for current transmission to end if
NOWAIT set. */

#ifndef Bufferdef
#define Bufferdef


typedef volatile char huge *BUF_PTR;

typedef struct
{
   BUF_PTR          BASE;       /* Base of buffer */
   BUF_PTR          TOP;        /* Top of buffer */
   unsigned long    SIZE;       /* Size of buffer */
   BUF_PTR          IN;         /* Input pointer of buffer */
   BUF_PTR          OUT;        /* Output pointer of buffer */
   volatile int     STATUS;     /* Status of buffer
NORM,EBUFFUL,EBUFEMPTY */
} BUFFER;

#endif


/**********************************************\
   Low level serial communication routines
\**********************************************/
/* Defines */
#define XON     10                      /* Serial transmission constants
*/
#define XOFF    20

/* Word Size */
#define BIT7    2                       /* Note:PH_IFX uses BIT8
only */
#define BIT8    3
/* Stop Bits */
#define STOP1   0                       /* Note:PH_IFX uses STOP1
only */
#define STOP2   4
/* Parity */
#define NONE    0                       /* Note:PH_IFX uses NONE
only */
#define ODD     8
#define EVEN    24
/* Baud Rate */
#define B1200   96                      /* Note:PH_IFX uses B1200 -
B9600 only */
#define B2400   48
#define B4800   24
#define B9600   12

#define BAUD_RATE   B9600               /* Define the baud rate */
#define PROTOCOL    BIT8|STOP1|NONE     /* Port protocol */

#ifndef BASEPORT
#error No valid COM port defined. Check SYSCONFG.H
#endif

#if (BASEPORT == COM1)
   #define COM_PORT     0
   #define COM_INT      0xC
   #define ENBL_IRQ     0xEF            /* Mask to turn on IRQ4 */
   #define MASK_IRQ     0x10            /* Mask to turn off IRQ4 */
   #define COM_PRIOR    0xC3            /* Makes COM1 highest priority int
*/
#endif

#if (BASEPORT == COM2)
   #define COM_PORT     1
   #define COM_INT      0xB
   #define ENBL_IRQ     0xF7            /* Mask to turn on IRQ3 */
   #define MASK_IRQ     0x08            /* Mask to turn off IRQ3 */
   #define COM_PRIOR    0xC2            /* Makes COM2 highest priority int
*/
#endif

/* 8250 UART registers */               /* Description */
#define BAUDLO      BASEPORT            /* lsb baud rate reg. */
#define DATAPORT    BASEPORT            /* transmit/receive data port */
#define BAUDHI      BASEPORT+1          /* msb baud rate reg. */
#define IER         BASEPORT+1          /* interrupt enable reg. */
#define IIR         BASEPORT+2          /* reason for interrupt */
#define LCR         BASEPORT+3          /* line control register */
#define MCR         BASEPORT+4          /* modem control register */
#define LSR         BASEPORT+5          /* line status register */
#define MSR         BASEPORT+6          /* modem status register */

/* 8250 values */
#define COMTIMEOUT  0x8000              /* Timeout (bit 15) returned by
bioscom */
#define DLAB        0x80                /* LCR. Enable divisor latch */
#define OUT2        0x08
/* MCR. Aux. output 2 latch (allows 8250 to send interrupts to system). */
#define XMITRDY     0x20                /* LSR. Able to send character */

/* 8250 interrupt enable values */
#define ENBL_DRDY   1                   /* data ready interrupt */
#define ENBL_TRDY   2                   /* transmit ready interrupt */
#define ENBL_LSTA   4                   /* line status change interrupt */
#define ENBL_MSTA   8                   /* modem status change interrupt */

/* 8250 interrupt causes */
#define INTMS       0                   /* int caused by modem status */
#define INTTX       2                   /* int caused by transmit ready */
#define INTRD       4                   /* int caused by receive ready */
#define INTLS       6                   /* int caused by line status */
#define MASK_IIR    0x6                 /* Mask off all but bits 1,2 of IIR */

/* 8259 ports and values */
#define IMR         0x21                /* Interrupt Mask Register (8259) */
#define IRR         0x20                /* Interrupt Control Register (8259) */
#define EOI         0x20                /* Non-specific End Of Interrupt */

#ifndef Structdef
```

```
#define Structdef
/* Typedefs */
typedef volatile union
{
   struct
   {
      DRDY : 1;
      TRDY : 1;
      LRDY : 1;
      MRDY : 1;
      REST : 4;
   } bit;
   unsigned char val;
} IER_BYTE;

typedef volatile union
{
   struct
   {
      Pending   : 1;
      ID        : 2;
      NotUsed   : 5;
   } bit;
   unsigned char val;
} IIR_BYTE;

typedef volatile struct
{
   int Send;      /* Transmit-if-able status */
   int Emerg;     /* Emergency type (if any) */
   int StatID;    /* Storage of status identifier */
   int MovCnt;    /* Number of complete moves buffered in SEND_BUFFER */
} COM_STATUS_STRUCT;

#endif

/* Enable_/Disable_ macros */
#define EnableT /* Data transmisssion */ \
   if (!ComStat.Emerg)                   \
   {                                     \
      ComStat.Send = ON;                 \
      IEReg.bit.TRDY = TRUE;             \
      outportb(IER,IEReg.val);           \
   }

#define DisableT /* Data transmission */ \
   {                                     \
      IEReg.bit.TRDY = OFF;              \
      outportb(IER,IEReg.val);           \
   }

#define EnableR /* Data reception */     \
   {                                     \
      IEReg.bit.DRDY = TRUE;             \
      outportb(IER,IEReg.val);           \
   }

#define DisableR /* Data reception */    \
   {                                     \
      IEReg.bit.DRDY = OFF;              \
      outportb(IER,IEReg.val);           \
   }

/* Emergency stop macro
#define Emerg_Stop(Err)                  \
   {                                     \
      ComStat.Emerg = Err;               \
      ComStat.Send = OFF;                \
      DisableT;                          \
      SendImmediately(EMERG_STOP_STRING); \
   }
*/
#define   Wait   while(kbhit())      \
                    getch();         \
                 while(!kbhit());
/*************************************\
   Prototypes and Global Variables
\*************************************/

/*************************************************************\
   (System Level) Low Level Serial Communications Routines
\*************************************************************/

int SendImmediately(char str[]);

/*********************\
   Level 1 Routines
\*********************/

/* Buffer Maintenance Routines */
extern BUF_PTR Pred(BUFFER *, BUF_PTR);
extern BUF_PTR Succ(BUFFER *, BUF_PTR);
extern int Buffer(CON_CODE, BUFFER *);
extern int ReadBuffer(lpCON_CODE, BUFFER *);
extern unsigned FreeBufSpace(BUFFER *);
extern unsigned UsedBufSpace(BUFFER *);

/* Character type checking routines */
extern int isdelim(CON_CODE);
extern int isXcode(CON_CODE);

/*********************\
   Level 2 Routines
\*********************/

/* Buffer Maintenance Routines */
extern BUFFER *Init_Buffer(unsigned);
extern void Flush_Buf(BUFFER *);
extern int StrBuffer(char Str[], BUFFER *);
extern int StrReadBuffer(char Str[], BUFFER *);
extern int _StrReadBuffer(char Str[], BUFFER *);

/* Status Structure Management */
extern int IDStat(char Str[]);
extern int GetStatus(int,char Str[], long);

/* Miscellaneous */
extern int Init_CheckIFX(void);
extern int Init_IFX(void);

/*********************\
   Level 3 routines
\*********************/

extern int Init_Robot(void);

/************************\
   Interface functions
\************************/
extern int GenCtrlCode(lpACTCOORD,char CtrlCode[]);
extern int SendCtrlCode(char CtrlCode[]);
extern int DrvHome(float);
extern int EnableM(long);

/*********************\
   Global Variables
\*********************/

extern COM_STATUS_STRUCT ComStat;
extern IER_BYTE IEReg;
extern BUFFER *SEND_BUFFER; /* Pointer to send buffer */
```

F.8 Ph_ifx.drv

```
/*************************************************************************\
 PH_IFX.DRV
 Motor driver for robot control programs.
 ========================================

 PARKER-HANNIFIN Digiplan IFX motor driver.

 This does not need to be explicitly included in the robot control
 program. It should be specified in the Motor Driver section of
 SYSCONFG.H.
\*************************************************************************/

//#define DEBUG

#include <bios.h>
#include <conio.h>
#include <ctype.h>
#include <dos.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <math.h>
#include "SYSCONFG.H"
#include GEOM_SOLN_HEADER    /* Header file for geometric solution */
#include "USER.H"            /* User level defines and prototypes. */
#include "PH_IFX.H"

/**********************\
   Function Prototypes
\**********************/
#if defined DEBUG
FILE *DebugFile;
#endif

/* System Level Routines */
int Init_Com(void);
void interrupt Com_Int_Handler(void);

/* Level 1 Routines */
/* Buffer Maintenance Routines */
BUF_PTR Pred(BUFFER *, BUF_PTR);
BUF_PTR Succ(BUFFER *, BUF_PTR);
int Buffer(CON_CODE, BUFFER *);
int ReadBuffer(lpCON_CODE, BUFFER *);
unsigned FreeBufSpace(BUFFER *);
unsigned UsedBufSpace(BUFFER *);

/* Character Type Checking Routines */
int isdelim(CON_CODE);
int isXcode(CON_CODE);

/* Level 2 Routines */
/* Buffer Maintenance Routines */
BUFFER *Init_Buffer(unsigned);
void Flush_Buf(BUFFER *);
int StrBuffer(char Str[], BUFFER *);
int StrReadBuffer(char Str[], BUFFER *);
int _StrReadBuffer(char Str[], BUFFER *);

/* Status Structure Management */
//int IDStat(lpCON_CODE);
int GetStatus(int,char Str[], long);

/* Miscellaneous */
int Init_CheckIFX(void);
int Init_IFX(void);
int GenXcode(lpACTCOORD Motor_vel, long *Incr_steps, char CtrlCode[]);
int CalcArmVel(lpACTCOORD, long *, float);

/* Level 3 Routines */
int Init_Robot(void);
atexit_t Exit_Robot(void);

/* Interface functions */
//int GenCtrlCode(lpACTCOORD, lpCON_CODE[]);
//int SendCtrlCode(lpCON_CODE[]);
int DrvHome(float);
int EnableM(long);

/* Global variables for buffer routines */
BUFFER *SEND_BUFFER;                                                    /* Pointer to send buffer */
BUFFER *BS_STRING;                                                      /* Pointer to dummy buffer */
BUFFER *STAT_BUFFER[MAX_STATUS*MOTOR_NUMBER+MOTOR_NUMBER];              /* Pointer to status receive buffer */

int STAT_BUFFER_SIZE[MAX_STATUS] = { 3,     /* B    */
                                     10,    /* BS   */
                                     12,    /* IS   */
                                     3,     /* R    */
                                     3,     /* RA   */
                                     3,     /* RB   */
                                     3,     /* RC   */
                                     3,     /* RS   */
                                     4,     /* TS   */
                                     10,    /* W3   */
                                     2,     /* XSR  */
                                     9,     /* FR   */
                                     12,    /* PR   */
                                     8,     /* PX   */
                                     18,    /* RV   */
                                     3,     /* STM  */
                                     4,     /* XC   */
                                     2,     /* XSD  */
                                     2,     /* XSP  */
                                     2 };   /* XSS  */

int MotorOffset[MOTOR_NUMBER] = MOTOR_OFFSETS;

/* Global variables for communications routines */
IER_BYTE IEReg;
IIR_BYTE IIReg;
COM_STATUS_STRUCT ComStat = { NORM, NORM, NO_STAT , 0};
const CON_CODE BS_String[5] = {BC_BS,'1','B','S',' '};

void interrupt (*COM_VECT) (); /* Old COM port interrupt vector */
/* Emergency stop */
void Emerg_Stop(int Err)
{
        ComStat.Emerg = Err;
        ComStat.Send = OFF;
```

```
        DisableT;
        SendImmediately(EMERG_STOP_STRING);
//      printf("\nError %d",Err);
}

/*************************************\
   System Level Routines (Level 0)
\*************************************/
/*************************************************************************\
   Low level serial communication routines
   (Only COM1 & COM2 are implemented)
\*************************************************************************/

/* This routine initializes both the 8250 COM chip registers and the 8259 interrupt controller registers.*/
int Init_Com()
{
   disable();

   /* Set up interrupt vector */
   COM_VECT = getvect(COM_INT);                          /* Gets interrupt vector. Returns   a 4 byte value to be stored in the   */
                                                         /* interrupt vector COM_VECT.                                            */
   setvect(COM_INT,Com_Int_Handler);                     /* Sets interrupt vector entry. Com_Int_Handler points to a function     */
                                                         /* that will be called when interrupt COM_INT occurs                     */
   outportb(MCR,inportb(MCR) | OUT2);                    /* Set up 8250 to generate interrupt requests to int controller          */
                                                         /* read MCR, bitwise OR with OUT2 (8bin), when bit 3 of MCR is set       */
                                                         /* high the connection between the OUT2 pin on the 8250 and the IRQ3     */
                                                         /* pin on the 8259 is enabled                                            */
   outportb(IER,NULL);                                   /* Initially no interrupts enabled at 8250.                              */

   /* Set port protocol and baud rate */
   outportb(LCR, DLAB);                  /* Set DLAB for BAUD access*/  /* Set the divisor latch access bit in the LCR register high  */
   outportb(BAUDLO, BAUD_RATE & 0xFF);   /* send lsb BAUD_RATE */       /* thus changing the multifunction registers 0x3F8 and 0x3F9 to */
   outportb(BAUDHI, (BAUD_RATE>>8) & 0xFF); /* send msb BAUD_RATE */    /* set baud rate mode. Then send appropriate bits to registers.*/

   outportb(LCR,PROTOCOL);                               /* Set data bits, stop bits and parity and also reset DLAB to            */
                                                         /* original value.                                                       */

   IIReg.val = inportb(IIR);                             /* Clear any pending interrupts from 8250                                */

   /* While bit 0 of the IIR register is low clear pending interrupts */
   while (!IIReg.bit.Pending)
   {
      switch(abs(IIReg.bit.ID))   /* Switch on bits 1 & 2 (Id of active */
      {                           /*  interrupt) of IIR                 */
         case 0: inportb(MSR);              break;
         case 1: outportb(DATAPORT,0);      break;
         case 2: inportb(DATAPORT);         break;
         case 3: inportb(LSR);              break;
      }
      IIReg.val = inportb(IIR);   /* Reread value of IIR */
   }

   outportb(IRR,COM_PRIOR);                              /* Re-prioritise interrupts so that Com port has the highest priority    */

   outportb(IMR,inportb(IMR) & ENBL_IRQ);                /* Set up interrupt controller to recognise interrupts. Read Interrupt   */
                                                         /* mask register, set bit 3 or 4 (IRQ3/4) low -> on                      */

   enable();

   /* Get value of IER register for later use */
   IEReg.val = inportb(IER);

   return(EXIT_SUCCESS);
}

/* void interrupt Com_Int_Handler - Hooked to the communications interrupt (IRQ3 or 4). Places received characters in specified buffers
   STAT_BUFFER[buffer Id] and sends characters from the SEND_BUFFER. Also disables sending if BC_PAUSE is encounterd or if a Status command is sent. */

   static struct {            /* Simulated Xon/Xoff variables */
                  int Request;
                  int OnOff;
                  int Cnt;
                 } X = { XON, XON, 0 };

void interrupt Com_Int_Handler()
{
   register unsigned char Ch;
   BUFFER *Buf;

   disable();
   sound(110);

   IIReg.val = inportb(IIR);

   do
   {
      switch(IIReg.val & MASK_IIR                        /* Switch on either a receive or transmit interrupt                      */
      {
         case INTRD :                                    /* Receive a character                                                   */
         sound(110);


         Buf = STAT_BUFFER[ComStat.StatID];              /* Use specific buffer to store received characters                      */
         Ch = inportb(DATAPORT);                         /* Get character from com port                                           */
         switch (Ch)                                     /* Switch on received character                                          */
         {
            case ' '    :                                /* If character is a space then reset Buffer IN pointer back to the BASE */
               Buf->IN = Buf->BASE;                      /* of the buffer. That is - clear the buffer because what we have        */
```

```
            break;                                          /* received so far is garbage.                          */

        case '\r'   :                                       /* All status replies from the IFX are terminated with a \r, so if \r is  */
                                                            /* received the previous string of characters should be a status reply.   */
            if (ComStat.StatID == 1*MOTOR_NUMBER) /* Buf. status req */  /* Check to see if a BC_BS ( IFX buffer size ) status command had been    */
                {                                           /* sent. If so test the most significant character to see how much IFX    */
                                                            /* buffer space has been used to the nearest hundred.                    */
                switch ( *Buf->BASE ) /* switch on MSB is 1BS reply */
                {
                    case '0'    :                           /* IFX Buffer nearly full so request sending to be temporily stopped by  */

                    case '1'    :                           /* the transmit part of the Com_Int_Handler. Note-this is only a request */

                    case '2'    :
                        X.Request = XOFF;                   /* for  the sending to be stopped, as it is possible only part of a      */
                        break;                              /* command has been sent. This will   allow sending to continue until the */
                                                            /* entire command has been sent                                          */

                    /*case '2'  : */                        /* IFX Buffer okay, do not request any change in sending                 */
                    case '3'    :
                    /*  break;  */

                    case '4'    :                           /* IFX Buffer nearly empty. Request that sending restarted               */
                    case '5'
                        X.Request = XON;
                        break;
                    default:                                /* Unexpected condition                                                  */
                        Emerg_Stop(600);
                }
            }

            Buf->STATUS = EBUFFUL;                          /* Reset buffer for reading                                              */
            Buf->OUT = Buf->BASE;
            ComStat.StatID = NO_STAT,                       /* If a \r has been received, reset the current status ID and disable    */
            DisableR;                                       /* reception.                                                            */

            if(ComStat.Send == ON)                          /* If no BC_PAUSE has been encountered enable transmission               */
                EnableT;
            /* Note: no break, puts \r in buffer */
        default    :                                        /* Store char in specific buffer specified by ComStat.StatID at top of   */
                                                            /* this routine.                                                         */
            *Buf->IN++ = Ch;
            if(Buf->IN > Buf->TOP)
            {
                Buf->IN = Buf->BASE;
            }

            break;
    }
    break,    /* Receive a character */

    case INTTX : /* Transmit a character */

    switch (X.OnOff)                                        /* This switch is needed to ensure that an entire command is sent. So    */
    {                                                       /* even if sending has been requested off (IFX buffer less than 100      */
                                                            /* free) sending will continue until the complete command has been sent. */

        case XON    :                                       /* Continue to send                                                      */
            Buf = SEND_BUFFER;
            if (Buf->STATUS == EBUFEMPTY)
            {
                Emerg_Stop(ESENDBUFMT);
                break;
            }
            Ch = *Buf->OUT++;
            Buf->OUT = (Buf->OUT > Buf->TOP ? Buf->BASE : Buf->OUT);

            /* Set the buffer status word */
            Buf->STATUS = (Buf->OUT == Buf->IN ? EBUFEMPTY : NORM);

            if (Ch == ' ' && X.Request == XOFF)             /* If the character just sent about to be sent is a space (therefore     */
                X.OnOff = XOFF;                             /* must be at end of command) and sending has been requested off, set    */
                break;                                      /* flag X.OnOff to XOFF ie sending of characters in SEND_BUFFER will be  */
                                                            /* temporarily stopped                                                   */

        case XOFF   :
            Buf = BS_STRING;                                /* If sending of characters from SEND_BUFFER has been stopped continue   */
            Ch = BS_String[X.Cnt++];                        /* to send 1BS to check on free space in IFX buffer.                     */

            if (Ch == ' ')                                  /* If sending has been requested ON ie enough space in IFX buffer and a  */
            {                                               /* space has just been sent (ie entire 1BS command has been sent), set   */
                if (X.Request == XON)                       /* flag X.OnOff to XON ie resume sending. If not space do not set flag   */
                    X.OnOff = XON;                          /* ON so that rest of 1BS command can be sent                            */
                    X.Cnt = 0;
            }
            break;

        default    :
            Emerg_Stop(700);
    }

            if ((int)Ch >= BC_STAT_BASE)                    /* If Ch is a BC code (other than BC_PAUSE) send a space in its place    */
            {                                               /* and turn on reception to read reply                                   */
//              ComStat.StatID = ((int)Ch - BC_STAT_BASE)*MOTOR_NUMBER;/* Calculate the status structure index for this status command. */
//              + *Buf->OUT - 0x31;                         /* 0x31 ASCII code for "1"                                               */
                ComStat.StatID = 3; // Always set for 1BS
                inportb(DATAPORT);
                EnableR;
```

```
            outportb(DATAPORT,' ');
        }

        else
        {
            switch (Ch)                                  /* If Ch is a BC_PAUSE, stop transmission (set flag ComStat.Send to   */
            {                                            /* OFF). If Ch is a normal movement command character, send character */
                case BC_PAUSE :
                    ComStat.Send = OFF;
                    DisableT;
                    break;

                default    :
                    outportb(DATAPORT,Ch);
            }
        }

        if ((ComStat.StatID != NO_STAT) && (Ch == ' '))  /* If flag ComStat.StatID is currently set to a status (ie a status   */
            DisableT;                                    /* command is to be sent)and Ch is a space, stop transmission until a */
                                                         /* reply has been received (ie \r). This stops multiple status commands */
                                                         /* being sent.                                                        */
        break;              /* Transmit a character */

      default:
        Emerg_Stop(800);   /* Unexpected condition */
    }

    IIReg.val = inportb(IIR);                            /* Recheck IIR to see if any transmit or receive interrupts have      */
                                                         /* occurred whilethis routine was being executed. While this rechecking */
                                                         /* would not seem to be strictly necessary, since, as soon as this    */
                                                         /* routine is finished the new interrupt should cause this routine to be */
                                                         /* invoked again, some characters were occasionaly lost.              */
} while (!IIReg.bit.Pending);

outport(IRR,EOI);                                        /* Send end-of-interrupt to interrupt controller                      */
enable();
nosound();
}

/* This routine SendImmediately() enables a string of charaters to be sent
   immediately to the IFX controller by, bypassing the SEND_BUFFER */
int SendImmediately(char Str[])
{
    int index;

    /* Temporarily disable COM interrupts */
    DisableT;

    for (index = 0; index != strlen(Str); index++)
    {
        while (!(inportb(LSR) & XMITRDY));   /* Wait until UART ready */
        disable();
        outportb(DATAPORT,*(Str + index));
        enable();
    }

    /* Re-enable COM interrupts */
    disable();
    outportb(IER,IEReg.val);
    enable();
    nosound();
    return(EXIT_SUCCESS);
}


/*********************\
   Level 1 Routines
\*********************/

/*****************************************************************************\
   Buffer maintenance routines
\*****************************************************************************/

/* Init_Buffer(unsigned) - Initialises the specified BUFFER structure.*/

BUFFER *Init_Buffer(unsigned siz)
{
    BUFFER *Buf;

    Buf = malloc(sizeof(BUFFER));       /* Allocates space for buffer       */
    if (Buf == NULL)
        return(NULL);

    Buf->BASE = malloc(siz);
    if (Buf->BASE == NULL)
    {
        free(Buf);
        return(NULL);
    }                                   /* Buffer pointers initialized      */

    Buf->TOP = Buf->BASE + siz - 1;     /* Pointer to top of buffer         */
    Buf->SIZE = siz;
    Buf->IN = Buf->BASE;                /* IN is at base of buffer          */
    Buf->OUT = Buf->BASE;               /* OUT is at base of buffer         */
    Buf->STATUS = EBUFEMPTY;            /* Buffer is statue is set empty    */

    return(Buf);
}
```

```
/* Pred(BUFFER *, pointer *) - returns pointer to predecessor of pointer in specified buffer. */
BUF_PTR Pred(BUFFER *Buf, BUF_PTR Ptr)
{
    return(--Ptr < Buf->BASE ? Buf->TOP : Ptr);
}

/* Succ(BUFFER *, pointer *) - returns pointer to successor of pointer in specified buffer. */
BUF_PTR Succ(BUFFER *Buf, BUF_PTR Ptr)
{
    return(++Ptr > Buf->TOP ? Buf->BASE : Ptr);
}

/* Buffer(CON_CODE, BUFFER * ) - Places a single character in the specified buffer, and manipulates the status byte. Function returns the buffer status. */
int Buffer(CON_CODE Ch, BUFFER *Buf)
{
    /* Prevent interruption of buffering process by Com_Int_Handler causing the pointers to change unexpectedly. */
    disable();

    switch (Buf->STATUS)
    {
        case NORM       :
        case EBUFEMPTY  :
            (*Buf->IN++ = Ch);
            if ((Buf == SEND_BUFFER) && (Ch == BC_PAUSE))
                ComStat.MovCnt++;
            Buf->IN = (Buf->IN > Buf->TOP ? Buf->BASE : Buf->IN);
            break;

        case EBUFFUL    :
            return(EBUFFUL);

        default         :
            return(EFAULT);
    }

    /* Store status, NORM or EBUFEMPTY */
    Buf->STATUS = (Buf->IN == Buf->OUT ? EBUFFUL : NORM);

    enable();

    return(NORM);
}

/* ReadBuffer(CON_CODE, BUFFER *) - Reads a single character from the specified buffer and manipulates the status byte. Function returns the status byte. */
int ReadBuffer(lpCON_CODE Ch, BUFFER *Buf) /* Reads characters from a buffer */
{
    if (Buf == SEND_BUFFER)
        return(ESENDBUF); /* Cannot use ReadBuffer with SEND_BUFFER */

    disable();  /* Prevent interruption of debuffering process by Com_Int_Handler causing the pointers to change unexpectedly. */
    switch (Buf->STATUS)
    {
        case NORM       :
        case EBUFFUL    :
            *Ch = *Buf->OUT++;          /* Get char from buffer, increment pointer */
            Buf->OUT = (Buf->OUT > Buf->TOP ? Buf->BASE : Buf->OUT);
            break;

        case EBUFEMPTY  :
            return(EBUFEMPTY),

        default
            return(EFAULT);
    }

    /* Store status, NORM or EBUFEMPTY */
    Buf->STATUS = (Buf->IN == Buf->OUT ? EBUFEMPTY : NORM);

    enable();

    return(NORM);
}

/* UsedBufSpace(BUFFER *) - Calculates (and returns) the used space in the
        specified buffer. */

unsigned UsedBufSpace(BUFFER *Buf)
{
    unsigned long Used,InSeg,InOff,OutSeg,OutOff;

    disable();
    InSeg = (unsigned long)FP_SEG(Buf->IN);
    InOff = (unsigned long)FP_OFF(Buf->IN);
    OutSeg = (unsigned long)FP_SEG(Buf->OUT);
    OutOff = (unsigned long)FP_OFF(Buf->OUT);
    enable();
    Used = (InSeg<<4) + InOff - (OutSeg<<4) - OutOff;

    /* Used = IN - OUT note: if IN < OUT ie IN pointer wrapped around then Used = (IN - OUT) + Buf->SIZE.
        When the final character fills the Buffer exactly UsedBufSpace returns an incorrect value (0) */
    Used += ((signed)Used < 0 ? Buf->SIZE:0),

    return((unsigned)Used);
}
```

```
/* FreeBufSpace(BUFFER *) - Calculates (and returns) the free space in the specified buffer. */
unsigned FreeBufSpace(BUFFER *Buf)
{
    unsigned Used;
    Used = UsedBufSpace(Buf);
    return((unsigned)((unsigned long)Buf->SIZE - Used));
}

/* StrBuffer(lpCON_CODE, BUFFER *) - buffers an entire string except for \x0. */
int StrBuffer(char Str[], BUFFER *Buf)
{
    int i,Len;

    Len = strlen(Str);
    if ((Len+1) > FreeBufSpace(SEND_BUFFER))
    {
        EnableM(NULL);
        return(EBUFFUL);
    }

    for (i = 0; i != Len; i++)
    {
        Buffer(*(Str + i),Buf);
        if (Buf->STATUS)
        return(Buf->STATUS);
    }

    return(EXIT_SUCCESS);
}

/* StrReadBuffer(lpCON_CODE, BUFFER *) - Reads a (delimiter terminated) string from the specified buffer, over-writing the contents of the specified string. */
int StrReadBuffer(char Str[], BUFFER *Buf)
{
    strcpy(Str, "");
    return(_StrReadBuffer(Str, Buf));
}

/* _StrReadBuffer(lpCON_CODE, BUFFER *) - Reads a (delimiter terminated) string from the specified buffer, appending to the contents of the specified string. */
int _StrReadBuffer(char Str[], BUFFER *Buf)
{
    int RetCode;
    CON_CODE Ch;

    do
    {
        if (!(RetCode = ReadBuffer(&Ch, Buf)))
            strncat(Str,&Ch,1);
    } while(!isdelim(Ch) && (RetCode != EBUFEMPTY));

    return(RetCode);
}

/* Flush_Buf(BUFFER *Buf) -flushes a buffer. */
void Flush_Buf(BUFFER *Buf)
{
    Buf->IN = Buf->BASE;        /* IN is at base of buffer */
    Buf->OUT = Buf->BASE;       /* OUT is at base of buffer */
    Buf->STATUS = EBUFEMPTY;  /* Set STATUS to empty */
}


/*********************************\
    Status Structure Management
\*********************************/

/* IDStat(lpCON_CODE, int *) - Identifies whether a specified string is an X Code status command and gives it an identifying status number that
    can be used as the index in the status information array. If the string is not a valid status command then the function returns MAX_STATUS */

lpCON_CODE STATUS_COMMAND_LIST[MAX_STATUS] =    /* Immediate status commands      */
                                            {   "B",    /* Buffer status          */
                                                "BS", /* Buffer size status       */
                                                "IS", /* Input status             */
                                                "R",  /* Ready request            */
                                                "RA", /* Limit status request     */
                                                "RB", /* Multifunction status     */
                                                "RC", /* Go home status           */
                                                "RS", /* Report seq status        */
                                                "TS", /* Trigger status           */
                                                "W3", /* Immed position report    */
                                                "XSR",/* Sequence run status      */

                                                /* Buffered status commands       */
                                                "FR", /* Encoder func report      */
                                                "PR", /* Position report          */
                                                "PX", /* Encoder posit report     */
                                                "RV", /* Software revision req    */
                                                "STM",/* Set power on defaults    */
                                                "XC", /* Comp EEPROM checksum     */
                                                "XSD",/* Download status          */
                                                "XSP",/* Power on run mode req    */
                                                "XSS"}; /* Sequence status        */
```

```
int IDStat(char Str[])
{
    int i = 0,c = 0;                                    /* STATUS_COMMAND_LIST index and character */
    int n,Dig;                                          /* Backtrack counter, Leading digit */

    strtok(Str,DELIMITERS);
    Dig = atoi(Str++);                                  /* Get first char, increment pointer */
                                                        /* Check that first char is a valid digit */
    if ((Dig == 0) || (Dig > MOTOR_NUMBER))
        return(MAX_STATUS*MOTOR_NUMBER);

    while ((i != MAX_STATUS) &&
           !((\0' == Str[c]) && (\0' == STATUS_COMMAND_LIST[i][c])))
        if (Str[c] == STATUS_COMMAND_LIST[i][c])
            c++;
        else
        {
            for (n = 0; n != c; n++)
                if (Str[n] != STATUS_COMMAND_LIST[i][n])
                {
                    c = 0;
                    break;
                }
            i++;
        }
    /* Convert i to a status structure index */
    i = MOTOR_NUMBER * i + (Dig - 1);

    return(i);
}

/* GetStatus - reads the status structure for status reply. If no reply is available it will buffer a request for the status information. Arg 1 is
the status ID number (same as that returned by IDStat).
Arg 2 is a pointer to the return string.
Arg 3 is a bitfield parameter that may contain:
NOASK   - do not ask controller if status buffer is empty (Overrides DOASK).
DOASK   - ask controller even if status buffer has data.
NOWAIT- do not wait for reply if asking controller.
NULL - reply if status buffer full, ask controller if not and wait for reply.
The function returns EBUFEMPTY if no reply is available (or NOWAIT
specified). It also returns EXIT_SUCCESS if reply extracted or ENOTSTATUS if a bad status structure index was passed. */

int GetStatus(int StID,char Str[], long Param)
{
    int Retcode = EXIT_SUCCESS,StName,StNum;
    CON_CODE StCode;
    char SRq[10];
    BUFFER *Buf = STAT_BUFFER[StID];

    StName = div(StID,MOTOR_NUMBER).quot;
    StNum  = div(StID,MOTOR_NUMBER).rem + 1;
    if ((StID >= MAX_STATUS*MOTOR_NUMBER) || (StID < 0))
        return(ENOTSTATUS);
    if (((Retcode = StrReadBuffer(Str,Buf)) == EBUFEMPTY) || (Param & DOASK))
        if (!(Param & NOASK))
        {
            StCode = BC_STAT_BASE + StName;
            sprintf(SRq,"%c%d%s %c",StCode,StNum,STATUS_COMMAND_LIST[StName],BC_PAUSE);
            if(Retcode = StrBuffer(SRq,SEND_BUFFER))
                return(Retcode);
            Flush_Buf(Buf);
            EnableM(NULL);
            if (!(Param & NOWAIT))
            {
                while(ComStat.Send != OFF);
                StrReadBuffer(Str,Buf);
            }
            else
                Retcode = EBUFEMPTY; /* Indicates no reply available */
        }
    return(Retcode);
}

/*************************************\
    Character type checking routines
\*************************************/

/* isdelim(CON_CODE) - returns TRUE if character is a delimiter space or \r. */
int isdelim(CON_CODE Ch)
{
    if (strchr(DELIMITERS,Ch))
        return(TRUE);
    return(FALSE);
}

/* isXcode(CON_CODE) -checks whether char is a valid Xcode character. */
int isXcode(CON_CODE Ch)
{
    if (isalnum(Ch) || isdelim(Ch) ||
        (Ch == '+') || (Ch == '-') || (Ch == '.'))
        return(TRUE),
    return(FALSE);
}
```

```
/*************************************************************************\
    Miscellaneous routines
\*************************************************************************/

/* Initial_checkIFX() sends 1RV to the IFX controller. The response is representative of the version of software used in the IFX controller.
    The reply is compared to the expected reply 'Str3'. If the characters are the same EXIT_SUCCESS is returned */

int Init_CheckIFX()
{
    int Recv;
    CON_CODE Str1[10],Str2[17],Str3[17];
    struct time start;
    struct time now;
    float starttm,nowtm;

    /* Clear status struct to hold returned chars */
    Recv = IDStat("1RV");
    Flush_Buf(STAT_BUFFER[Recv]);

    Flush_Buf(SEND_BUFFER);

    sprintf(Str1,"* %c1RV %c",BC_RV,BC_PAUSE);
    strcpy(Str2, "");

    if (StrBuffer(Str1,SEND_BUFFER))
        return(ESENDBUF);

    EnableM(NULL);

    /* Wait for reply to be received or if don't receive reply in less than IFXTIMEOUT then break out and return IFX not ready    */
    gettime(&start);
    starttm = start.ti_sec + start.ti_hund/100;

    while(ComStat.Send != OFF)
    {
        gettime(&now);
        nowtm= now.ti_sec + now.ti_hund/100;
        if(nowtm - starttm >= IFXTIMEOUT)
        {
            DisableT;
            ComStat.Send = OFF;
            ComStat.StatID = NO_STAT;
            Flush_Buf(SEND_BUFFER);
            return(EIFXNOTREADY);
        }
    }

    StrReadBuffer(Str2, STAT_BUFFER[Recv]);

    strcpy(Str3,IFX_VERSION);                       /* Our version of IFX software */

    if (strncmp(Str2,Str3,strlen(Str3)))
        return(EIFXBADREPLY);


    /* Flush Buffers to flush 1RV from SEND_BUFFER and reply from STAT_BUFFER[RV]    */
        Flush_Buf(SEND_BUFFER);

    return(EXIT_SUCCESS);
}

/* Init_IFX - initialises the motor controller */
int Init_IFX(void)
{
    CON_CODE Init_String[CTRLCODE_LENGTH];

    sprintf(Init_String,   "MN 1A%d 2A%d 3A%d 1V0 2V0 3V0 1D0 2D0 3D0 G O11 ",MAX_MOTOR_ACC,MAX_MOTOR_ACC,MAX_MOTOR_ACC);
    return(StrBuffer(Init_String,SEND_BUFFER));
}

/* CalcArmVel - Calculates all arm velocities. If the maximum motor velocity is exceeded, it returns ETOOFAST and uses the maximum velocity. */
int CalcArmVel(lpACTCOORD Motor_vel, long *Incr_steps, float WayTm)
{
    int i,Retcode = EXIT_SUCCESS;
    SPACECOORD D;
    float TA,Dmax,b,l,h,r,Rad,Largest = 0;

    for(i=0;i<MOTOR_NUMBER;i++) /* Find largest distance */
    {
        D[i] = fabs((float)Incr_steps[i]/MOTOR_RES);
        Largest = (D[i] > Largest ? D[i] : Largest);
    }
    /* Calc area of trapezium 1/2(b-l)h+hl */
    b = WayTm;
    l = ((l=b - 2*RISETIME)>0 ? l : 0);
    h = ((WayTm>2*RISETIME) ? MAX_MOTOR_VEL : MAX_MOTOR_ACC * WayTm/2);

    Dmax = 0.5 * (b-l)*h + l*h;
    if (Dmax < Largest)                     /* Cannot move distance in time requested */
    {
        Retcode = ETOOFAST;

        /* Recalculate time */
        r = Largest - RISETIME*MAX_MOTOR_VEL;                   /* rectangular area */
        WayTm = (r > 0 ? 2*RISETIME + r/MAX_MOTOR_VEL :         /* Trapezoid */
                    sqrt(Largest/MAX_MOTOR_ACC)*2 );            /* Triangle */
    }
```

```
    TA = WayTm * MAX_MOTOR_ACC;

    /* Calculate motor velocities */
    for(i=0;i<MOTOR_NUMBER;i++)
    {
        if ((Rad = TA*TA - 4*MAX_MOTOR_ACC*D[i]) < 0)
            Rad = 0;
        Motor_vel[i] = (TA - sqrt(Rad))/2;
    }
    return(Retcode);
}
/* GenXcode - generates the command string */
int GenXcode(lpACTCOORD Motor_vel, long *Incr_steps,char CtrlCode[])
{
    static int Trig = 0,Outp;

    Trig = ((Trig == 1) ? 0 : 1);
    Outp = 1 - Trig;

#if defined DELTA
    sprintf(CtrlCode, "1V%2.2f 2V%2.2f 3V%2.2f 1D%ld 2D%ld 3D%ld G "
                      "O%d%d TRX%d%d %c1BS ",
                      Motor_vel[0],Motor_vel[1],Motor_vel[2],
                      Incr_steps[0],Incr_steps[1],Incr_steps[2],
                      Outp,Outp,Trig,Trig,BC_BS);

#elif defined KIWI
    sprintf(CtrlCode, "1V%2.2f 2V%2.2f 3V%2.2f 1D%ld 2D%ld 3D%ld G "
                      "O%d%d TRX%d%d %c1BS ",
                      Motor_vel[0],Motor_vel[1],Motor_vel[2],
                      Incr_steps[0],Incr_steps[1],Incr_steps[2],
                      Outp,Outp,Trig,Trig,BC_BS);

#elif defined CANTTRK
    sprintf(CtrlCode, "1V%2.2f 2V%2.2f 1D%ld 2D%ld G "
                      "O%d%d TRX%dX %c1BS ",
                      Motor_vel[0],Motor_vel[1],
                      Incr_steps[0],Incr_steps[1],
                      Outp,Outp,Trig,BC_BS);
#endif

    return(EXIT_SUCCESS);
}

/*********************\
    Level 3 routines
\*********************/

/* This routine Init_Robot initializes the memory buffers and their
    respective pointers. It also checks that communication is open with the
    IFX controller. */
int Init_Robot()
{
    int RetCode,i,j,index;

    /* Initialise the SEND_BUFFER and other pointers */
    if (!(SEND_BUFFER = Init_Buffer(SEND_BUFFER_SIZE)))
        return(ENOMEM);
    if (!(BS_STRING = Init_Buffer(1)))
        return(ENOMEM);
    Buffer('1',BS_STRING);

    /* Initialise status structure */
    for (i = 0; i < MAX_STATUS; i++)
        for (j = 0; j < MOTOR_NUMBER; j++)
        {
            index = i*MOTOR_NUMBER + j;
            if (!(STAT_BUFFER[index] = Init_Buffer(STAT_BUFFER_SIZE[i])))
                return(ENOMEM);
        }

    if (RetCode = Init_Com())
        return(RetCode);

    /* All initialisations successful if this point is reached */
    if (RetCode = atexit((atexit_t) Exit_Robot))
        return(RetCode);

    /* This routine is a checking procedure for presence of the IFX controller. */
    /*  if (RetCode = Init_CheckIFX())
        return(RetCode);*/

    /* Buffer initialisation code for IFX */
    if (RetCode = Init_IFX())
        return(RetCode);

    return(EXIT_SUCCESS);
}

/* This routine atexit_t Exit_Robot() is called when ever the program is  exited. It disables the the communication interrupts IRQ3 or IRQ4
 in the 8259. It disables the connection between the Com chip pin OUT2 and the 8259. It resets the interrupt vector entry. */

atexit_t Exit_Robot()
{
    disable();

    /* Un-init COM settings */
    outportb(IMR,inportb(IMR) | MASK_IRQ);          /* Disable IRQ3 or 4, 8259      */
```

```
    outportb(IER,0);                            /* clear all interrupt enables*/
    outportb(MCR,inportb(MCR) & ~OUT2);         /* disable out2 on 8250       */
    setvect(COM_INT,COM_VECT);

    enable();
    return(EXIT_SUCCESS);
}

/* GenCtrlCode - Takes actuator coordinates and converts them to controller commands to move to the appropriate position. */
int GenCtrlCode(lpACTCOORD Arm_angles,char CtrlCode[])
{
    int i, Retcode;
    int Mtr_steps[MOTOR_NUMBER];
    long Incr_steps[MOTOR_NUMBER];
    float WayTm;
    ACTCOORD Motor_steps;
    ACTCOORD Motor_vel;

    /* Convert arm angles to motor steps */
    for(i=0;i<MOTOR_NUMBER;i++)
    {
        Motor_steps[i] = Arm_angles[i]*GEAR_RATIO*MOTOR_RES/(2*Pi)-MotorOffset[i];

        /* Convert motor steps to a whole number */
        Mtr_steps[i] = round(Motor_steps[i]);

        /* Calculate steps between waypoints */
        Incr_steps[i] = Mtr_steps[i] - Robot.motor[i];

        /* Store new position in Robot structure */
        Robot.motor[i] = Mtr_steps[i];
    }

    /* Generate Xcode */
    WayTm = Robot.way_dist/GetSpeed();

    /* Calculate arm velocities */
    Retcode = CalcArmVel(Motor_vel, Incr_steps, WayTm);

    /* Generate Xcode */
    GenXcode(Motor_vel, Incr_steps, CtrlCode);
    return(Retcode);
}

/* SendCtrlCode - interface function to programmer module */
int SendCtrlCode(char CtrlCode[])
{
    int RetCode;
    while(((RetCode = StrBuffer(CtrlCode,SEND_BUFFER)) == EBUFFUL));

/*
#if defined DEBUG
DebugFile=fopen("debug.dat","a+");
fprintf(DebugFile,"%s\n",CtrlCode);
fclose(DebugFile);
#endif
*/
    return(RetCode);
}

/* DrvHome - Homes the robot */
int DrvHome(float Rate)
{
    int Retcode;
    CON_CODE CtrlCode[20];

    if (Rate>MAX_HOME_RATE)
        Rate = MAX_HOME_RATE;
    sprintf(CtrlCode," S GH%2.2f %c",Rate,BC_PAUSE);
    if (Retcode = StrBuffer(CtrlCode,SEND_BUFFER))
        return(Retcode);
    return(EnableM(NULL));
}

/* EnableM - enables a buffered move */
int EnableM(long Param)
{
    int Retcode;

    /* Wait until latest move is finished */
    while (ComStat.Send == ON)
    {
        if (Param & NOWAIT)         /* if NOWAIT specified, then exit */
            return(ENOENABLE);
    }

    if (ComStat.MovCnt < 0)             /* Check that MovCnt is not corrupt */
        return(EFAULT);

    if (ComStat.MovCnt == 0)        /* Ensure that BC_PAUSE is buffered */
        if (Retcode = Buffer(BC_PAUSE,SEND_BUFFER))
            return(Retcode);

    ComStat.MovCnt--;
    EnableT;

    return(EXIT_SUCCESS);
}
```

F.9 User.h

```
/*************************************************************************\
   USER.H
   High level robot control routines.
   ==================================
   This does not need to be explicitly included  in the robot control
   program, although it must be present or the program will not compile.
   See ROBOT.H for more information.
\*************************************************************************/
/************\
   Defines
\************/

/* Errors */
#undef EBASE
#define EBASE           EBASEUSER
#define EBADSPEED       EBASE+1     /* SetSpeed has been sent bad value */
#define EBADMODE        EBASE+2     /* SetSpeedMode has been sent bad value */

#define ESINGULAR       EBASE+3     /* Mathematical equation is singular */

/* Mode specifiers */
#define CARTESIAN       1           /* Coordinate system modes */
#define SPHERICAL       2
#define TIME_MODE       1           /* Speed specification mode */
#define VELO_MODE       2

#define COLINEARITY_TOL 0.0001      /* Used in PathCirc corresponds to ~0.9
                                       degrees divergence between vectors */


/************\
   Typedefs
\************/
#ifndef RobotStructDef

#define RobotStructDef

typedef struct {
   int coord;
   int SpeedMode;
   float  speed;
   float accel;
   float way_dist;
   int motor[MOTOR_NUMBER];
   float endpt[SPACEDIMEN];
} ROBOTSTRUCT;
#endif


/**************\
   Prototypes
\**************/
/* Robot Structure Management */
extern int SetSpeed(float);
extern float GetSpeed(void);
extern int SetEndPt(lpSPACECOORD);
extern int GetEndPt(float [SPACEDIMEN]);

/* Movement Functions */
extern int _Move(SPACECOORD);
extern int Home(float);
extern int Move(lpSPACECOORD, float);
extern int Path(SPACECOORD, float);
extern int KiwiHoldPos(float Rate);
extern int CantHoldPos(float Rate);
extern int PathCirc(lpSPACECOORD, lpSPACECOORD, float, float);
extern int PathtrkK(float,float,float,float,float,float,float);

/* Math, Matrix & Vector Functions */
extern int round(double);
extern void conv_angle_rads(float *);
extern void AddVector(float[3], float[3], float[3]);
extern void SubtractVector(float[3], float[3], float[3]);
extern void CrossprodVec(float [3],float [3],float [3]);
extern void DotProdVector(float [3],float [3],float *);
extern void Calc_UnitVec(float [3], float [3]);
extern void MagVec(float[3], float *);
extern void MultScalVec(float, float[3], float[3]);
extern void MultMatVec3(float M[3][3],float B[3],float V[3]);
extern void MultMatVec4(float M[4][4],float B[4],float V[4]);
extern void MultMatMat(float A[3][3],float B[3][3],float C[3][3]);
extern void MatTrans(float M[3][3],float I[3][3]);

/********************\
   Global variables
\********************/
extern ROBOTSTRUCT Robot;
```

F.10 User.c

```
/************************************************\
/* User.c */
/************************************************/
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "SYSCONFG.H"
#include GEOM_SOLN_HEADER
#include "USER.H"
#include MOTOR_DRIVER

#include "ph_ifx.h"

/* Prototypes */

/* Robot structure management */
int SetSpeed(float);
float GetSpeed(void);
int SetEndPt(lpSPACECOORD Pt);
int GetEndPt(SPACECOORD Pt);

/* Movement functions */
int _Move(SPACECOORD);
int Move(lpSPACECOORD, float);
int Home(float);
int KiwiHoldPos(float Rate);
int CantHoldPos(float Rate);
int Path(SPACECOORD, float);
int PathCirc(lpSPACECOORD, lpSPACECOORD, float, float);
int PathtrkK(float,float,float,float,float,float,float);

/* Math, Matrix & Vector functions */
int round(double);
void conv_angle_rads(float *);
void AddVector(float[3], float[3], float[3]);
void SubtractVector(float[3], float[3], float[3]);
void CrossprodVec(float [3],float [3],float [3]);
void DotProdVector(float [3],float [3],float *);
void Calc_UnitVec(float [3], float [3]);
void MagVec(float[3], float *);
void MultScalVec(float, float[3], float[3]);
void MultMatVec3(float M[3][3],float B[3],float V[3]);
void MultMatVec4(float M[4][4],float B[4],float V[4]);
void MultMatMat(float A[3][3],float B[3][3],float C[3][3]);
void MatTrans(float M[3][3],float I[3][3]);

/* Global variables */
ROBOTSTRUCT Robot = ROBOTSTRUCT_INIT;

/*****************************\
   Robot structure management
\*****************************/

/* SetSpeed - Places argument 1 in Robot.speed */
int SetSpeed(float Speed)
{
   if (Speed <= MINSPEED)
      return(EBADSPEED);
   Robot.speed = Speed;

   return(EXIT_SUCCESS);
}

/* GetSpeed - returns current endpoint speed in m/s */
float GetSpeed(void)
{
   return(Robot.speed);
}

/* SetEndPt - Sets spatial endpoint position (Robot.endpt) to value of argument 1 */
int SetEndPt(lpSPACECOORD Pt)
{
   int i;

   for(i=0; i<SPACEDIMEN; i++)
     Robot.endpt[i] = Pt[i];

   return(EXIT_SUCCESS);
}

/* GetEndPt - Copies the current endpoint (Robot.endpt) into first argument */
int GetEndPt(SPACECOORD Pt)
{
   int i;

   for(i=0; i<SPACEDIMEN; i++)
     Pt[i] = Robot.endpt[i];

   return(EXIT_SUCCESS);
}
```

```
/* SetMotor - Set the actuator position (Robot.motor) to value of arg. 1 */
int SetMotor(ACTCOORD Pt)
{
    int i;

    for(i=0; i<MOTOR_NUMBER; i++)
        Robot.motor[i] = Pt[i];

    return(EXIT_SUCCESS);
}

/* GetMotor - Copies the current motor position (Robot.motor) into first argument. */
int GetMotor(ACTCOORD Pt)
{
    int i;

    for(i=0; i<MOTOR_NUMBER; i++)
        Pt[i] = Robot.motor[i];

    return(EXIT_SUCCESS);
}


/**********************\
    Movement functions
\**********************/
/* _Move        - Generates and buffers movement code but does not enable movement
    Mtr_steps   - motor steps for each arm as a whole number.
    Incr_steps  - number of motor steps for each arm between way points.
    Time        - time between way points. Normally Time = Robot.speed.time,
                  but may be increased for the particular way point being
                  processed in CalcArmVel() if it is too short).
    Arm_angles  - angles made by each arm for a particular platform position.
    Motor_steps - same as Mtr_steps but a float ie can take nos smaller than
                  one so must be converted to whole number for IFX.
    Motor_vel   - calculated velocity of each arm during move between way points. */

int _Move(SPACECOORD Pt)
{
    int Retcode = EXIT_SUCCESS,Retcode2;
    ACTCOORD Arm_angles;
    CON_CODE CtrlCode[CTRLCODE_LENGTH];

    /* Call kinematics routine */
    if((Retcode = Invkin(Pt,Arm_angles))!=0)
    {
        return(Retcode);
    }

    /* Generate controller code */
    Retcode = GenCtrlCode(Arm_angles,CtrlCode);

    /* Buffer code */

    Retcode2= SendCtrlCode(CtrlCode);
    while(Retcode2 !=0 ){
        return(Retcode2);
    }

    return(Retcode);
}

/* Move - Moves the robot to given point. */
int Move(lpSPACECOORD Pt, float Speed)
{
    int Retcode;
    float temp_way,dist;
    SPACECOORD sp,Vec;

    if (Speed)
        SetSpeed(Speed);

    /* Temporarily set way_ dist to length of the move for call to _Move */
    temp_way = Robot.way_dist;
    GetEndPt(sp);
    SubtractVector(Pt,sp,Vec);
    MagVec(Vec,&dist);
    Robot.way_dist = dist;

    switch(Retcode = _Move(Pt))
    {
        case EXIT_SUCCESS:
        case ETOOFAST:
            break;
        case EBUFFUL:
            ComStat.MovCnt++;
            EnableM(NULL);
            break;
        default:
            return(Retcode);
    }

    Robot.way_dist = temp_way;
    SetEndPt(Pt);
    return(EnableM(NULL));
}

/* Home() - returns robot to home position */
int Home(float Rate)
{
    float Pt[SPACEDIMEN] = HOME_ENDPT;
    float Mt[MOTOR_NUMBER] = MOTR_ENDPT;
    int Retcode;

    /* Drive robot into safe position for homing operation */
    if (Retcode = KinHome(Rate))
        return(Retcode);

    /* Call motor controllers inbuilt homing functions or implement
        alternative homing procedure */
    if (Retcode = DrvHome(Rate))
        return(Retcode);

    /* Home procedure successful */
    SetEndPt(Pt);
    SetMotor(Mt);

    return(Retcode);
}

/* KiwiHoldPos() - returns Kiwibot to a holding position */
int KiwiHoldPos(float Rate)
{
    float Pt[SPACEDIMEN] = KIWIHOLD_PT;
    float Mt[MOTOR_NUMBER] = MOTR_ENDPT;
    int Retcode;

    printf("\n\nPress any key to return robot to holding position.");
    Wait;

    /* Drive robot into holding position */
    Move(Pt,Rate);

    EnableM(NULL);

    /* Holding proceedure successful */
    SetEndPt(Pt);
    SetMotor(Mt);

    return(Retcode);
}

/* CantHoldPos() - returns Kiwibot to a holding position */
int CantHoldPos(float Rate)
{
    float Pt[SPACEDIMEN] = CANTHOLD_PT;
    float Mt[MOTOR_NUMBER] = MOTR_ENDPT;
    int Retcode;

    printf("\n\nPress any key to return robot to holding position.");
    Wait;

    /* Drive robot into holding position */
    Move(Pt,Rate);

    EnableM(NULL);

    /* Holding proceedure successful */
    SetEndPt(Pt);
    SetMotor(Mt);

    return(Retcode);
}

// Path - moves endpoint in spatial path.
int Path(SPACECOORD Pt, float Speed)
{
    float PathVec[3],PathVec2[3],WayPt[3],StartPt[3];
    float PathMag,f,df;
    int Retcode;

    /* Set speed*/
    if(Speed)
        SetSpeed(Speed);

    StartPt[0] = Robot.endpt[0];
    StartPt[1] = Robot.endpt[1];
    StartPt[2] = Robot.endpt[2];

    SubtractVector(Robot.endpt,Pt,PathVec);
    MagVec(PathVec, &PathMag);
    df = Robot.way_dist/PathMag;

    f = 0;
```

```
    do
    {
        if ((f += df) > 1)
            f = 1;
        MultScalVec(f,PathVec,PathVec2);
        AddVector(StartPt,PathVec2, WayPt);

        switch(Retcode = _Move(WayPt))
        {
            case EXIT_SUCCESS:
            case ETOOFAST:
                SetEndPt(WayPt);
                break;
            case EBUFFUL:
                EnableM(NULL);
                break;
            default:
                return(Retcode);
        };

    }while(f< 1);

    SetEndPt(WayPt);

    return(EnableM(NULL));
}

/* PathCirc - constructs a circular path using the current position as
    starting point and arg 1 as the centre. The vector W (arg 2) defines the
    orientation of the circle's plane. Arg 3 is the arc angle of the path
    and the last argument is the endpoint speed.
    The component of W that is normal to the line between the centre C and
    the current endpoint is used to define the orientation of the circle
    plane. For example if W = {0, 0, 1} then the circle will lie in the XY
    plane. If the normal component is near to zero then the plane is
    undefined and an error is returned. */
int PathCirc(lpSPACECOORD C, lpSPACECOORD W, float Ang, float Speed)
{
    float Ainc,A;
    float R,Dp,a,b;
    float U[3],unitU[3],V[3],unitV[3],unitW[3],aU[3],bV[3],
          P1[3],WayPt[3],S[3];
    int Retcode;

    /* Set speed */
    if(Speed)
        SetSpeed(Speed);

    GetEndPt(S);
    conv_angle_rads(&Ang);
    SubtractVector(C,S,U);
    MagVec(U,&R);
    if (R == 0)
        return(EXIT_SUCCESS); /* Zero radius move */

    Ainc = Robot.way_dist/R;

    Calc_UnitVec(U,unitU);
    Calc_UnitVec(W,unitW);

    /* Check for incompatible orientation vector W */
    DotProdVector(unitW,unitU,&Dp);
    if (fabs(Dp) > 1 - COLINEARITY_TOL)
        return(ESINGULAR);

    CrossprodVec(unitW,unitU,V);
    Calc_UnitVec(V,unitV);

    A=0;

    do
    {
        if((A += Ainc )> Ang)
        A = Ang;

        a = R*cos(A);
        b = R*sin(A);

        MultScalVec(a,unitU,aU);
        MultScalVec(b,unitV,bV);

        AddVector(aU,bV,P1);
        AddVector(P1,C,WayPt);

        switch(Retcode = _Move(WayPt))
        {
            case EXIT_SUCCESS:
            case ETOOFAST:
                SetEndPt(WayPt);
                break;
            case EBUFFUL:
                EnableM(NULL);
                break;
            default:
                return(Retcode);
        },

    } while(A<Ang);

    return(EnableM(NULL));
}

/* PathtrkK - constructs a fictitious orbital path for the tracked object
        Lamdai   - initial starting angular offset (degs)
        Lamda    - angle of rotation of tracked object (degs)
        Radius   - radius of circular path (m)
        Height   - distance of origin of circle from base (m)
        Omega    - angular velocity of tracked object (degs/s)
        Alpha3   - 3 angle (z axis) setting orientation of circle (degs)
        Alpha2   - 2 angle (y axis) (degs) */

int PathtrkK(float Lamdai,float Lamda,float Radius,float Height,float Omega,float
Alpha3,float Alpha2)
{
    float C_x4xI[3][3],C_xIx4[3][3],C_x5x4[3][3],C_x4x5[3][3],
          C_xIx5[3][3],C3theta3a[3][3],C3theta3ainv[3][3],C2theta2[3][3],
          C2theta2inv[3][3],C_temp[3][3],C3[3][3],P_x4[3],P_x5[3],P[3],
          theta3a,theta2,Ang,Anginc,Lamdaf,Time,WayTm,WayPt[3],Speed,prevAng;

    int Retcode;
    static int movecnt = 1;

    conv_angle_rads(&Lamdai);
    conv_angle_rads(&Lamda);
    conv_angle_rads(&Alpha3);
    conv_angle_rads(&Alpha2);
    conv_angle_rads(&Omega);

    /* Angular increments of platform (rads) */
    Anginc = Robot.way_dist*Pi/180;

    /* Orientation matrix of circle */
    C_x4xI[0][0] = cos(Alpha3)*cos(Alpha2);
    C_x4xI[0][1] = -sin(Alpha3);
    C_x4xI[0][2] = cos(Alpha3)*sin(Alpha2);
    C_x4xI[1][0] = sin(Alpha3)*cos(Alpha2);
    C_x4xI[1][1] = cos(Alpha3);
    C_x4xI[1][2] = sin(Alpha3)*sin(Alpha2);
    C_x4xI[2][0] =  -sin(Alpha2);
    C_x4xI[2][1] = 0;
    C_x4xI[2][2] = cos(Alpha2);

    /* Position of tracked object expressed in x5y5z5 system */
    P_x5[0] = Radius;
    P_x5[1] = 0;
    P_x5[2] = Height;

    /* Calc final angular position */
    Lamdaf = Lamda + Lamdai;

    WayTm =0;
    Ang = -Anginc;
    Time = 0;

    while(Ang<Lamdaf)
    {
        prevAng = Ang;

        if((Ang+=Anginc)>Lamdaf)
        {
            Ang = Lamdaf;       // Occur when angular increments are not divisiable into
Lamdaf

            // Calc new reduced angular increment
            Anginc = Lamdaf - prevAng;
        }

        /* Set speed to medium for first move i.e. move to start of tracking position */
        if(movecnt == 1)
        {
            SetSpeed(MEDIUM);
            Speed = 0;
        }
        else
        {
            if(Omega)
            {
                SetSpeed(Omega*180/Pi);
                Speed = Omega;
            }
        }

        /* Calc time between way points */
        if(movecnt>1)
        {
            WayTm = Anginc/Speed;
        }

        Time+=WayTm;

        C_x5x4[0][0] = cos(Lamdai + Speed*Time);
        C_x5x4[0][1] = -sin(Lamdai + Speed*Time);
        C_x5x4[0][2] = 0;
        C_x5x4[1][0] = sin(Lamdai + Speed*Time);
        C_x5x4[1][1] = cos(Lamdai + Speed*Time);
        C_x5x4[1][2] = 0;
```

```
        C_x5x4[2][0] = 0;
        C_x5x4[2][1] = 0;
        C_x5x4[2][2] = 1;

        /* Express position of tracked object in x4y4z4 frame */
        MultMatVec3(C_x5x4,P_x5,P_x4);

        /* Position of tracked object expressed in XYZ frame (note P expressed in inertial
    frame, no subscript) */
        MultMatVec3(C_x4xI,P_x4,P);

        /* Aiming angles in XYZ frame (determine which quadrant) */
        theta3a = atan(fabs(P[1]/P[0]));
        if(P[0] >= 0 && P[1] >= 0)
            theta3a = theta3a;
        else if(P[0] < 0 && P[1] >= 0)
            theta3a = Pi - theta3a;
        else if(P[0] < 0 && P[1] < 0)
            theta3a = Pi + theta3a;
        else
            theta3a = 2*Pi - theta3a;

        if(P[2] == 0)               /* stop devide by zero */
            theta2 = 90*Pi/180;
        else
            theta2 = atan(sqrt(pow(P[0],2) + pow(P[1],2))/fabs(P[2]));

        if(P[2] >= 0)
            theta2 = theta2;
        else
            theta2 = Pi - theta2;

        WayPt[0] = 0;               // theta3b not used
        WayPt[1] = theta2;
        WayPt[2] = theta3a;

        if(movecnt == 2)
        {
            printf("\n\nPress any key to start tracking.");
        Wait;
        }

        switch(Retcode = _Move(WayPt))
        {
            case EXIT_SUCCESS:
            case ETOOFAST:
                SetEndPt(WayPt);
                break;
            case EBUFFUL:
                EnableM(NULL);
                break;
            default:
                return(Retcode);
        };

        if(movecnt == 1)
        {
            printf("\nPress any key to move to start of tracking position.");
            Wait;
            EnableM(NULL);
        }
        movecnt++;
    };
    return(EnableM(NULL));
}


/************************************\
    Math, Matrix and vector functions
\************************************/

/* Rounds 1st argument and returns rounded number as an int */
int round(double Num)
{
    double Rem,Flr;

    Flr = floor(Num);
    Rem  = Num - Flr;
    if (Rem >= 0.5)
        return((int)ceil(Num));
    else
        return((int)Flr);
}

void conv_angle_rads(float *A)
{
    *A = (Pi/180.0) * (*A);
}

void MultMatMat(float A[3][3],float B[3][3],float C[3][3])
{
  int i,j,k,l,m,n;
  float temp;

    for(m=i=0; m<3 || i<3; m++, i++)
        for(n=l=0; n<3 || l<3; n++, l++){
            temp = 0;

            for(j=k=0; j<3 || k<3; j++, k++){
                temp  = temp + A[i][j] * B[k][l];
                C[m][n] = temp;
            }
        }
}

void MatTrans(float M[3][3],float I[3][3])
{
  I[0][0] = M[0][0];
  I[0][1] = M[1][0];
  I[0][2] = M[2][0];

  I[1][0] = M[0][1];
  I[1][1] = M[1][1];
  I[1][2] = M[2][1];

  I[2][0] = M[0][2];
  I[2][1] = M[1][2];
  I[2][2] = M[2][2];
}

/* Calc_UnitVec - converts argument 1 into a unit vector and returns
    the unit vector in argument 2. If all elements of arg 1 are zero then
    the return vector is the zero vector. */
void Calc_UnitVec(float V[3], float unitV[3])
{
    float Length;
    Length = sqrt(pow(V[0],2) + pow(V[1],2) + pow(V[2],2));
    if (Length == 0)
        Length = 1; /* Returns zero if Length is zero */
    unitV[0] = V[0]/Length;
    unitV[1] = V[1]/Length;
    unitV[2] = V[2]/Length;
}

Void MultScalVec(float Scal,float B[3],float V[3])
{
    V[0] = Scal * B[0];
    V[1] = Scal * B[1];
    V[2] = Scal * B[2];
}

void AddVector(float A[3],float B[3],float C[3])
{
    C[0] = B[0] + A[0];
    C[1] = B[1] + A[1];
    C[2] = B[2] + A[2];
}

void SubtractVector(float A[3],float B[3],float C[3])
{
    C[0] = B[0] - A[0];
    C[1] = B[1] - A[1];
    C[2] = B[2] - A[2];
}

void DotProdVector(float A[3],float B[3],float *Scal)
{
    *Scal = A[0]*B[0] + A[1]*B[1] + A[2]*B[2];
}

void CrossprodVec(float A[3],float B[3],float C[3])
{
    C[0] = A[1]*B[2] - A[2]*B[1];
    C[1] = A[2]*B[0] - A[0]*B[2];
    C[2] = A[0]*B[1] - A[1]*B[0];
}

void MagVec(float V[3],float *MagV)
{
    *MagV = sqrt(pow(V[0],2) + pow(V[1],2) + pow(V[2],2) );
}

void MultMatVec3(float M[3][3],float B[3],float V[3])
{
    V[0] = M[0][0] * B[0]+
           M[0][1] * B[1]+
           M[0][2] * B[2];

    V[1] = M[1][0] * B[0]+
           M[1][1] * B[1]+
           M[1][2] * B[2];

    V[2] = M[2][0] * B[0]+
           M[2][1] * B[1]+
           M[2][2] * B[2];
}
```

```
void MultMatVec4(float M[4][4],float B[4],float V[4])
{
    V[0] = M[0][0] * B[0]+
           M[0][1] * B[1]+
           M[0][2] * B[2]+
           M[0][3] * B[3];

    V[1] = M[1][0] * B[0]+
           M[1][1] * B[1]+
           M[1][2] * B[2]+
           M[1][3] * B[3];

    V[2] = M[2][0] * B[0]+
           M[2][1] * B[1]+
           M[2][2] * B[2]+
           M[2][3] * B[3];

    V[3] = M[3][0] * B[0]+
           M[3][1] * B[1]+
           M[3][2] * B[2]+
           M[3][3] * B[3];
}
```

F.11 Sysconfg.h

```
/*************************************************************************\
    SYSCONFG.H
    System configuration file for robot control programs.
    ======================================================

    This does not need to be explicitly included  in the robot control
    program, although it must be present or the program will not compile.
    See ROBOT.H for more information.
\*************************************************************************/

/*************************************************************************\
    Define the correct robot type. Options are:
    <Top of list>
    #define DELTA
    #define KIWI
    <End of list>
    One of these lines should appear as the next line after this
    comment.
\*************************************************************************/
#define DELTA
#if defined DELTA
    #define GEOM_SOLN_HEADER "DELTA.H"
#elif defined KIWI
    #define GEOM_SOLN_HEADER "KIWI.H"
#elif defined CANTTRK
    #define GEOM_SOLN_HEADER "CANTTRK.H"
#else
    #error Missing or Invalid robot definition. Use #define DELTA (or KIWI)
#endif

/*************************************************************************\
    Define the computer port configuration.
\*************************************************************************/
#define COM1        0x3f8
#define COM2        0x2f8

/*************************************************************************\
    Define the port that the motor controller is on. Options are:
    COM1
    COM2
\*************************************************************************/
#define BASEPORT COM2

/*************************************************************************\
    Define the motor driver file to be used. Options are:
    <Top of list>
    PH_IFX - Parker Hannifin Compumotor IFX
    <End of list>
\*************************************************************************/
#define MOTOR_DRIVER     "PH_IFX.H"

/*************************************************************************\
    Physical parameters of the robot.
    Check the motor driver header for the parameters that must be
    defined in this section.
\*************************************************************************/
#define MOTOR_RES        400    /* Resolution of the motors */
#define GEAR_RATIO       80     /* Ratio of gear reducers */
#define SLOW             0.1    /* A slow speed for (motor rev/s) for homing */
#define MEDIUM           0.5    /* A medium speed for homing */
#define FAST             1      /* A high speed for homing */
#define MAX_HOME_RATE    3      /* Fastest allowable homing speed */
#define MAX_MOTOR_VEL    50     /* Maximum allowable motor velocity */
#define MAX_MOTOR_ACC    35     /* Maximum allowable motor acceleration */

#if defined DELTA
    /* Offset from home position to kin. datum */
    /* Steps from horizontal to Delta home position
    Mt1 3055 Mt2 2990 Mt3 3004 */
    #define MOTOR_OFFSETS    {3055,2990,3004}
#elif defined KIWI
    /* Steps from Delta home position to Kiwi home position
    Mt1 5536 Mt2 5583 Mt3 5532
    Steps from vertical to Kiwi home position
    Mt1 5519 (62.09 deg) Mt2 5407 (60.83 deg) Mt3 5472 (61.56 deg) */
    #define MOTOR_OFFSETS    {5519,5407,5472}

#elif defined CANTTRK
    /* Steps from Delta home position to Kiwi home position
    Mt1 5536 Mt2 5583 Mt3 5532
    Steps from vertical to Kiwi home position
    Mt1 5519 (62.09 deg) Mt2 5407 (60.83 deg) Mt3 5472 (61.56 deg) */
    #define MOTOR_OFFSETS    {5519,5407,5472}
#else
    #error Missing or Invalid robot definition. Use #define DELTA (or KIWI)
#endif

    #define KIWIHOLD_PT {0,0,0}    // theta3a = 0 deg, theta2 = 0 deg (zenith)
    #define CANTHOLD_PT {0,0,0}    // theta3a = 0 deg, theta2 = 0 deg (zenith)


/*************************************************************************\
    Universal constants
\*************************************************************************/
#define OFF             0                   /* Bitwise OFF */
#define ON              -1                  /* Bitwise ON (two's compliment) */
#define FALSE           0                   /* Boolean OFF */
#define TRUE            1                   /* Boolean ON */
#define NORM            0                   /* Normal status value */
#define Pi              3.141592653589      /* or thereabouts */

/*************************************************************************\
    Error code bases
    These should not overlap. Number bases in steps of 100
\*************************************************************************/
#define EBASEUSER       100         /* User module */
#define EBASEDRIVER     200         /* Motor driver */
#define EBASEKIN        300         /* Kinematics module */
```

F.12 Robot.h

```
/*************************************************************************\
    ROBOT.H
    Header file for robot control programs.
    =======================================

    This is the only non-standard header file that needs to be included
    in the robot control program, although all the others provided must
    be present or the program will not compile.

    Files that must be present
    --------------------------
    ROBOT.H - This file
    SYSCONFG.H - System configuration information (user configurable)
    USER.H,USER.C - The high level robot commands module

    The correct driver for the motor:
    E.g. PH_IFX.DRV and any files specified in the title block of the
    driver (.DRV) file.

    The correct forward/inverse kinematics module:
    E.g. DELTA.KIN & DELTA.H and any files specified in the title block of
    the forward/inverse kinematics header.

\*************************************************************************/

/* Load the other header modules */
#include "SYSCONFG.H"
#include GEOM_SOLN_HEADER
#include "USER.H"
```